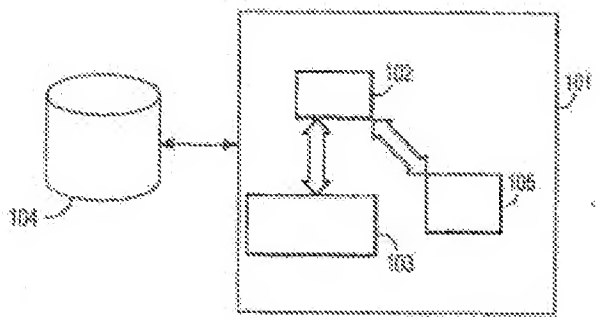
Espacenet

Bibliographic data: JP H11509664 (A)

VLIW PROCESSOR WHICH PROCESSES COMPRESSED INSTRUCTION FORMAT

Publication date:	1999-08-24	
Inventor(s):		
Applicant(s):		
Classification:	**- international:**	***G06F9/30; G06F9/318; G06F9/32; G06F9/38;*** (IPC1-7): G06F9/32; G06F9/38
	- European:	G06F9/30T; G06F9/30T2; G06F9/318T; G06F9/38E2; G06F9/38E6; G06F9/38T
Application number:	JP19970540698T 19970515	
Priority number (s):	WO1997IB00558 19970515; US19960648359 19960515; US19960649731 19960515; US19960649733 19960515	
Also published as:	• JP 3750821 (B2) • WO 9743710 (A2) • WO 9743710 (A3) • EP 0843846 (A2) • EP 0843846 (B1) • more	

Abstract not available for JP H11509664 (A)
Abstract of corresponding document: WO 9743710 (A2)



A VLIW processor uses a compressed instruction format that allows greater efficiency in use of cache and memory. Instructions are byte aligned and of variable length. Branch targets are uncompressed Format bits specify how many issue slots are used in a following instruction. NOPS are not stored in memory. Individual operations are compressed according to features such as whether they are resultless, guarded, short, zeroary, unary, or binary. Instructions are stored in compressed form in memory and in cache. Instructions are decompressed on the fly after being read out from cache.

Last updated: 12.10.2011 Worldwide Database 5.7.23.2; 92p

(19)日本国特許庁（ＪＰ）　(12) **公表特許公報（Ａ）**　(11)特許出願公表番号

特表平11－509664

(43)公表日　平成11年(1999)８月24日

(51)Int.Cl.[6]		識別記号	FI		
Ｇ０６Ｆ	9/38	３７０	Ｇ０６Ｆ	9/38	３７０Ｘ
	9/32	３５０		9/32	３５０Ａ

審査請求　未請求　予備審査請求　未請求(全 60 頁)

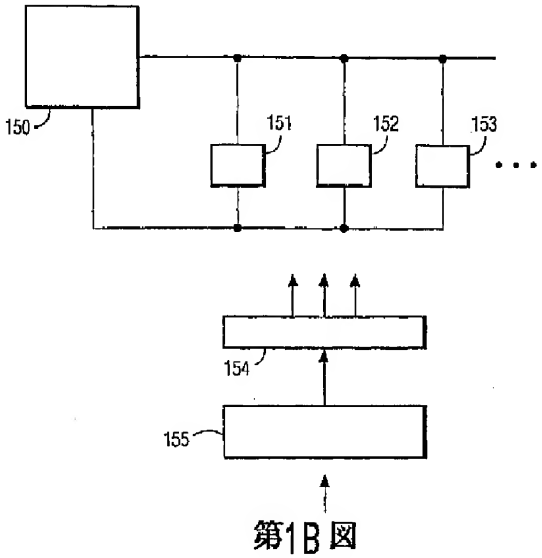
(21)出願番号　特願平9－540698
(86)(22)出願日　平成９年(1997)５月15日
(85)翻訳文提出日　平成10年(1998)１月14日
(86)国際出願番号　ＰＣＴ／ＩＢ９７／００５５８
(87)国際公開番号　ＷＯ９７／４３７１０
(87)国際公開日　平成９年(1997)11月20日
(31)優先権主張番号　０８／６４８，３５９
(32)優先日　1996年５月15日
(33)優先権主張国　米国（ＵＳ）
(31)優先権主張番号　０８／６４９，７３１
(32)優先日　1996年５月15日
(33)優先権主張国　米国（ＵＳ）

(71)出願人　フィリップス　エレクトロニクス　ネムローゼ　フェンノートシャップ
オランダ国　5621　ベーアー　アインドーフェン　フルーネヴァウツウェッハ　1
(72)発明者　ジェイコブス　エイノ
オランダ国　5656　アーアー　アインドーフェン　プロフ　ホルストラーン　6
(72)発明者　アング　マイケル
オランダ国　5656　アーアー　アインドーフェン　プロフ　ホルストラーン　6
(74)代理人　弁理士　沢田　雅男

最終頁に続く

(54)【発明の名称】　圧縮された命令フォーマットを処理するＶＬＩＷプロセッサ

(57)【要約】
本発明によるＶＬＩＷプロセッサは、キャッシュ及びメモリの使用効率が極めて高くなるような圧縮された命令フォーマットを用いる。命令はバイト位置合わせされた可変長のものである。分岐目標は圧縮されない。フォーマットビットは、後続の命令でどれだけ多くのイシュースロットが使用されるかを特定する。ノーオペレーションは上記メモリには記憶されない。個々の命令部は、それらが無結果のものであるか、保護されているか、短いか、零項のものであるか、単項のものであるか、又は２項のものであるか等の特徴に応じて圧縮される。命令は圧縮された形でメモリ及びキャッシュに記憶される。命令はキャッシュから読み出された後、即座に伸張される。



第1Ｂ図

【特許請求の範囲】

1．圧縮された命令を使用するＶＬＩＷプロセッサであって、当該プロセッサが、

－　複数のイシュースロットを有する命令イシューレジスタであって、各イシュースロットが各々の命令部を記憶するためのものであり、これら命令部の全てが同一のクロックサイクル内で実行を開始するような命令イシューレジスタと、

－　前記命令イシューレジスタに記憶された前記各命令部を実行するための複数の機能ユニットと、

－　伸張された命令を前記命令イシューレジスタに供給する伸張ユニットであって、当該伸張ユニットが圧縮命令記憶媒体から圧縮された命令を取り出すと共に該圧縮された命令を伸張し、前記圧縮された命令は各々が各圧縮命令部長さに圧縮されている少なくとも２個の命令部を含むような伸張ユニットと、

を有するようなＶＬＩＷプロセッサにおいて、

前記伸張ユニットは複数の有限長さから選択された各圧縮命令部長さで各々の命令部を伸張するように構成され、上記有限長さが少なくとも２個の非零長さを含んでいることを特徴とするＶＬＩＷプロセッサ。

2．請求項１に記載のプロセッサにおいて、前記伸張ユニットは前記圧縮命令記憶媒体からフォーマットフィールドを取り出すように構成され、該フォーマットフィールドは前記圧縮された命令の各命令部に対する各々の圧縮命令部長さを特定し、前記伸張ユニットが前記圧縮された命令の各命令部を上記フォーマットフィールドに従って伸張することを特徴とするＶＬＩＷプロセッサ。

3．請求項２に記載のプロセッサであって、前記フォーマットフィールドがＮ個の副フィールドを有し、Ｎがイシュースロットの数であり、各副フィールドが各イシュースロットに対する圧縮された命令部の長さを特定するようなプロセッサにおいて、

前記副フィールドの各々が少なくとも２ビットを含むことを特徴とするＶＬＩＷプロセッサ。

4．請求項２又は３に記載のプロセッサにおいて、前記伸張ユニットが、

－　前記圧縮命令記憶媒体から、先行する圧縮された命令を前記フォーマットフ

ィールドと共に取り出し、

－　前記先行する圧縮された命令の伸張を開始し、次いで、

－　前記圧縮命令記憶媒体から前記圧縮された命令を取り出すと共に、該圧縮された命令の伸張を、前記圧縮命令記憶媒体から前記先行する圧縮された命令と一緒に取り出された前記フォーマットフィールドに従って開始する、

ように構成されていることを特徴とするＶＬＩＷプロセッサ。

５．請求項２、３又は４に記載のプロセッサにおいて、前記圧縮ユニットはメモリアクセスユニット内の前記圧縮命令記憶媒体から前記フォーマットフィールドを取り出し、該メモリアクセスユニットは少なくとも１個の命令部副フィールドも有し、前記伸張ユニットは前記伸張された命令の前記命令部の少なくとも１つにおける前記命令部副フィールドを合成することを特徴とするＶＬＩＷプロセッサ。

６．圧縮された命令を使用するＶＬＩＷプロセッサであって、当該プロセッサが、

－　複数のイシュースロットを有する命令イシューレジスタであって、各イシュースロットが各々の命令部を記憶するためのものであり、これら命令部の全てが同一のクロックサイクル内で実行を開始するような命令イシューレジスタと、

－　前記命令イシューレジスタに記憶された前記各命令部を実行するための複数の機能ユニットと、

－　伸張された命令を前記命令イシューレジスタに供給する伸張ユニットであって、当該伸張ユニットが圧縮命令記憶媒体から圧縮された命令の流れを取り出すと共にこれら圧縮された命令を伸張し、前記命令の流れが命令圧縮フォーマットを特定するフォーマットフィールドを含む第１の命令を有しているような伸張ユニットと、

を有するようなＶＬＩＷプロセッサにおいて、

前記命令の流れは前記第１の命令に続いて前記圧縮命令記憶媒体から取り出される第２の命令を有し、前記伸張ユニットは前記第１の命令の前記フォーマットフィールドに従って前記第２の命令を伸張するように構成されていることを特徴とするＶＬＩＷプロセッサ。

７．ＶＬＩＷプロセッサ上で実行するための圧縮されたコードを生成する方法であって、当該方法が、

－　複数の命令部を有する命令を入力し、

－　前記命令の各命令部を、対応する圧縮命令部長さを当該命令部に割当てるような各圧縮方策に従って圧縮する、

ような各過程を有する方法において、

前記圧縮命令部長さが複数の有限の長さから選択され、これら有限の長さは少なくとも２つの非零長さを含み、これら有限長さの何れが選択されるかが当該命令部の少なくとも１つの特徴に依存することを特徴とする圧縮されたコードを生成する方法。

８．前記命令を含む命令の流れに適用される請求項７に記載の方法において、当該方法が、各命令に関して当該命令が前記命令の流れにおける他の命令からの分岐の分岐目標であるか否かを判定し、分岐目標でない命令のみを圧縮するような過程を有することを特徴とする圧縮されたコードを生成する方法。

９．請求項７又は８に記載の方法において、当該方法が、フォーマットフィールドを生成し、当該フォーマットフィールドが前記命令の各命令部に対する各フォーマットを当該命令部のために選択された前記圧縮命令部長さに従って特定するような過程を有していることを特徴とする圧縮されたコードを生成する方法。

１０．請求項９に記載の方法であって、前記フォーマットフィールドがＮ個の副フィールドを有し、Ｎはイシュースロットの数であり、各副フィールドが各イシュースロット用の圧縮命令部長さを特定するするような方法において、前記副フィールドの各々が少なくとも２ビットを含むことを特徴とする圧縮されたコードを生成する方法。

１１．請求項７、８、９又は１０に記載の方法において、前記圧縮された命令部を含む圧縮された命令を、電子計算機により読み取ることのできる圧縮命令記憶媒体に記憶する過程を有していることを特徴とする圧縮コードを生成する方法。

１２．請求項１１に記載の方法において、当該命令の実行に先立って実行するための他の命令を圧縮し、当該命令用の前記フォーマットが前記他の命令と共に

取り出すために記憶されることを特徴とする圧縮されたコードを生成する方法。

１３．請求項１１又は１２に記載の方法において、前記圧縮命令記憶媒体はメモリアクセスユニットを有し、前記フォーマットフィールドは前記命令の少なくとも１個の命令部の少なくとも１個の命令部副フィールドと共に同一のメモリに記憶されることを特徴とする圧縮されたコードを生成する方法。

１４．ＶＬＩＷプロセッサ上で実行するための圧縮されたコードを生成する方法であって、当該方法が、

－　各々が複数の命令部を有する命令の流れを入力し、

－　前記命令の各命令部を、対応する圧縮命令部長さを当該命令部に割当てるような各圧縮方策に従って圧縮し、

－　各命令に対してフォーマットフィールドを生成し、該フォーマットフィールドは当該命令の各命令部用の各フォーマットを当該命令部用に選択された前記圧縮命令部長さに従って特定し、

－　前記フォーマットフィールド及び命令を圧縮された形で記憶し、これらフォーマットフィールド及び命令の各々は各命令からの前記圧縮された命令部を電子計算機により読み取ることのできる圧縮命令記憶媒体中に含む、

ような各過程を有するような圧縮されたコードを生成する方法において、

前記命令の流れは当該流れからの第２の命令に先立って実行する第１の命令を有し、前記第２の命令に対応する前記フォーマットフィールドと前記第１の命令に対応する前記圧縮された命令は、前記第２の命令に対応する前記圧縮された命令の取り出しに先立ち、前記命令の流れの実行中に組み合わせて取り出されるように前記圧縮命令記憶媒体に記憶されることを特徴とする圧縮されたコードを生成する方法。

１５．請求項１４に記載の方法において、各命令に関して当該命令が前記命令の流れにおける他の命令からの分岐の分岐目標であるか否かを判定し、分岐目標でない命令のみを圧縮するような過程を有することを特徴とする圧縮されたコードを生成する方法。

１６．ＶＬＩＷプロセッサ上で実行するための圧縮されたコードを生成する方法であって、当該方法が、

－　各々が複数の命令部を有する命令の流れを入力し、

－　前記命令の各命令部を、対応する圧縮命令部長さを当該命令部に割当てるような各圧縮方策に従って圧縮し、

－　前記フォーマットフィールド及び命令を圧縮された形で記憶し、これらフォーマットフィールド及び命令の各々は各命令からの前記圧縮された命令部を電子計算機により読み取ることのできる圧縮命令記憶媒体中に含む、

ような各過程を有するような圧縮されたコードを生成する方法において、

当該方法は、各命令に関して当該命令が前記命令の流れにおける他の命令からの分岐の分岐目標であるか否かを判定し、分岐目標である各命令は圧縮されていない形で記憶し、分岐目標でない命令は圧縮された形で記憶するような過程を有することを特徴とする圧縮されたコードを生成する方法。

１７．請求項７ないし１６の何れか一項に記載の方法を実施するようにプログラムされた電子計算機。

【発明の詳細な説明】

圧縮された命令フォーマットを処理するＶＬＩＷプロセッサ

技術分野

本発明は、ＶＬＩＷ（超長命令語）プロセッサに係り、特に、そのようなプロセッサ用の命令フォーマット並びにそのような命令フォーマットを処理する装置及び方法に関する。

特に、本発明は請求項１の「において」なる語に先行する部分に規定したようなＶＬＩＷプロセッサに関する。

背景技術

ＶＬＩＷ命令を圧縮する方策は、米国特許第5,179,680号及び第5,057,837号で提案されている。この圧縮方策は、使用されない命令部（operations）が除去された命令語と、どの命令部が除去されたかを示すマスク語とを設けている。

ＶＬＩＷプロセッサは複数のイシュースロット（issue slots）を含む命令ワードを有している。これらプロセッサは複数の機能ユニットも含んでいる。各機能ユニットは、或る型式の一組の命令部を実行するためのものである。各機能ユニットは、一つの命令を各マシンサイクル中でパイプライン態様で開始することができる点でＲＩＳＣのようなものである。各イシュースロットは各命令部を保持するためのものである。同一命令ワード内の全ての命令部は、当該プロセッサの単一サイクル内において上記機能ユニット上で並列に開始されねばならない。このように、上記ＶＬＩＷは、きめの細かい並列処理をなす。

上記のように、ＶＬＩＷマシン上の一つの命令は典型的には複数の命令部を含んでいる。従来のマシン上では、各命令部は別個の命令として参照されていた。しかしながら、ＶＬＩＷマシンにおいては、各命令は複数の命令部又はノー・オペレーション（no-ops；ダミー命令部）からなっている。

従来のプロセッサと同様に、ＶＬＩＷプロセッサは、当該プロセッサ上で実行する命令の流れを記憶するためのディスクドライブのようなメモリ装置を使用する。又、ＶＬＩＷプロセッサは、従来のプロセッサのように、当該プロセッサに対する広帯域アクセス可能性を以て上記命令流の断片を記憶するためにキャッシ

ュを使用することもできる。

ＶＬＩＷマシン内の命令は、これらの命令部からプログラマ又はコンパイラにより構築される。このように、ＶＬＩＷプロセッサ内のスケジューリングはソフトウェアにより制御されるものである。

ＶＬＩＷプロセッサは、ベクトル・プロセッサ又はスーパースケーラ・プロセッサのような他の型式の並列プロセッサと比肩し得るものである。ベクトル・プロセッサは、複数データ項目に対して同時に実行される単一の各命令部を有している。スーパースケーラ・プロセッサはＶＬＩＷプロセッサのようにきめの細かい並列処理をなすが、ＶＬＩＷとは違って、ハードウェアで命令部をスケジュールする。

長い命令ワードのため、ＶＬＩＷプロセッサはキャッシュの使用に関して大きな問題を有している。特に、大きなコード寸法はキャッシュ・ミス、即ち必要な命令がキャッシュ内にないような状況、を引き起こす。大きなコード寸法は、又、主メモリからキャッシュへコードを転送するために主メモリの広帯域さが必要となる。

大きなコード寸法は下記のような要因により問題となり得る。

－　最適な実行のためにプログラムを細かく整合させるには、グラフティング、ループの展開、又はプロシージャの組み入れ（インライン）等の技術が使用される。これらの手順はコード寸法を増大させる。

－　各命令において、全てのイシュースロットは使用されていない。優良な最適化コンパイラは使用されていないイシュースロットの数を減少させることはできるが、或る数のノーオペレーション（ダミー命令）は殆どの命令流中に存在し続ける。

－　機能ユニットの使用を最適化するために、条件分岐に関する命令部は典型的には当該分岐遅延の経過前に、即ち何の分岐がとられようとしているか判る前に、開始される。何の結果が実際に使用されるべきかを決定するために、命令には保護ビットが含まれる。

－　より新しいプロセッサ型式で好適に使用される大きなレジスタファイルは、

一層長いアドレスを必要とし、これらアドレスは命令部と共に含まれねばならない。

ＶＬＩＷ命令の圧縮方策は米国特許第5,179,680号及び第5,057,837号で提案されている。この圧縮方策はマスクワードを用いて命令ワード中の使用されない命令部を削除するが、当該命令を圧縮する余地がもっとある。

本出願の技術背景についての更なる情報は、本明細書で参照する下記先行出願中に見られる：

－　１９９２年１２月２９日に出願された米国特許出願第９９８，０９０号（出願人整理番号：ＰＨＡ２１７７７）。該出願はきめ細かい並列処理をなすためのＶＬＩＷプロセッサ・アーキテクチャを示している；

－　１９９３年１０月２５日に出願された米国特許出願第１４２，６４８号（出願人整理番号：ＰＨＡ１２０５）。該出願は保護ビットの使用を示している；

－　１９９４年１２月３０日に出願された米国特許出願第３６６，９５８号（出願人整理番号：ＰＨＡ２１９３２）。該出願はＶＬＩＷアーキテクチャと共に使用されるレジスタファイルを示している。

プログラム圧縮技術の図書目録：

－　電子計算機及び通信システムのモデル解析及びシミュレーションに関する第２回国際ワークショップにおける、ジェー・ワン他による論文「メモリシステム性能を向上させる圧縮の使用の可能性」の第１０７～１１３頁（ダーハム、ＮＣ，米国１９９４年）；

－　並列計算、第１７巻、ｎ２～３、１９９１年６月の第２０７～２１９頁に記載のエッチ・シュローダ他による「命令収縮アレイ上のプログラム圧縮」；

－　ジェー・電子計算機及びソフトウェア技術、第２巻、第３番、第３１５～２７頁（１９９４年）に記載のエー・ウォルフ他による「埋め込みＲＩＳＣアーキテクチャ上での圧縮されたプログラムの実行」；

－　電子計算機の設計：電子計算機及びプロセッサにおけるＶＬＩＷについての１９９４年ＩＥＥＥ国際会議の原稿集における第２７０～７頁のエム・コズク他による「埋め込みシステムプログラムの圧縮」、（１９９４年１０月、１０～１

２、米国マサチューセッツ州ケンブリッジ）。

典型的には、これらの文献で採用されている取り組み方は、プログラムを全体として又はプログラムコードのブロックとして圧縮しようとするものである。更に、典型的には、これらの取り組み方では、命令位置又は命令のブロックの位置の何等かのテーブルが必要とされる。

発明の開示

本発明の目的は、ＶＬＩＷプロセッサにおけるコードの寸法を減少させることにある。

又、本発明の他の目的は、一層高度に圧縮された命令を処理するＶＬＩＷプロセッサを提供することにある。

本発明によるプロセッサは、前記伸張ユニットが複数の有限長さから選択された各圧縮命令部長さで各々の命令部を伸張するように構成され、上記有限長さが少なくとも２個の非零長さを含んでいることを特徴とする。利用できる命令部長さの上記組み合わせは、例えば０、２６、３４及び４２ビット長の圧縮された命令部である。どの命令部が特定の長さに圧縮されるかは、先ずこれら命令部の発生頻度の学習に依存する。これは、書かれたソフトウェアの形式に依存して変化し得る。更に、上記長さは当該命令部が保護されているか又は保護されていないかに、当該命令部が結果を生成するか否か、当該命令部が即値パラメータを使用するか否か、及び当該命令部が使用するオペランドの数に依存させることができる。

本発明によるプロセッサは、前記伸張ユニットが前記圧縮命令記憶媒体からフォーマットフィールドを取り出すように構成され、該フォーマットフィールドが前記圧縮された命令の各命令部に対する各々の圧縮命令部長さを特定し、前記伸張ユニットが前記圧縮された命令の各命令部を上記フォーマットフィールドに従って伸張するような実施例を有する。好ましくは、上記フォーマットフィールドは当該プロセッサのどのイシュースロットが当該命令により使用されるべきかも特定する。

本発明によるプロセッサの他の実施例であって、前記フォーマットフィールド

がＮ個の副フィールドを有し、Ｎがイシュースロットの数であり、各副フィールドが各イシュースロットに対する圧縮された命令部の長さを特定するようなプロセッサにおいては、前記副フィールドの各々が少なくとも２ビットを含むことを特徴とする。４つの異なる命令部長さが使用される場合は、上記副フィールドは例えば２ビット長であり得る。

本発明の他の実施例においては、前記伸張ユニットが、

－　前記圧縮命令記憶媒体から、先行する圧縮された命令を前記フォーマットフィールドと共に取り出し、

－　前記先行する圧縮された命令の伸張を開始し、次いで、

－　前記圧縮命令記憶媒体から前記圧縮された命令を取り出すと共に、該圧縮された命令の伸張を、前記圧縮命令記憶媒体から前記先行する圧縮された命令と一緒に取り出された前記フォーマットフィールドに従って開始する、

ように構成されている。このように、上記フォーマットフィールドは前記圧縮された命令がロードされる前に利用することができ、該フォーマットフィールドに従う伸張の準備を前記圧縮された命令がロードされる前に開始することができる。

本発明の一実施例においては、前記圧縮ユニットがメモリアクセスユニット内の前記圧縮命令記憶媒体から前記フォーマットフィールドを取り出し、該メモリアクセスユニットは少なくとも１個の命令部副フィールドも有し、前記伸張ユニットは前記命令部副フィールドを前記伸張された命令の前記命令部の少なくとも１つに合成する。この構成は取り出し効率を上昇させ、命令取り出しのパイプライン化を可能にする。この構成は、どのような数の利用可能な命令部長さ（例えば０及び３２なる２つの長さも）を伸張することができるプロセッサで使用することができる。このように、上記伸張ユニットは後続の命令にどのくらい多数のイシュースロットがあるかを当該命令を取り出すに先立ち報知される。分岐目標以外の各命令について、フォーマットを特定するフィールドが先行する命令と共に記憶される。

本発明は、請求項７によるＶＬＩＷプロセッサ上で実行するための圧縮されたコードを生成する方法にも関する。この方法は上記プロセッサにとって有用な命

令を発生する。

本発明による上記方法であって、当該方法が前記命令を含む命令の流れに適用されるような方法は、各命令に関して当該命令が前記命令の流れにおける他の命令からの分岐の分岐目標であるか否かを判定し、分岐目標でない命令のみを圧縮するような過程を有するような実施例を有する。このように、分岐目標は当該プロセッサによるプログラムの実行の間には伸張される必要はなく、分岐命令の実行の後に伸張するための遅延も必要とされない。

図面の簡単な説明

以下、本発明を例示により（本発明を限定するものではない）以下の図面を参照して説明する。

第１Ａ図は、本発明の圧縮命令フォーマットを使用するプロセッサを示す。

第１Ｂ図は、第１Ａ図のプロセッサにおけるＣＰＵの詳細を示す。

第２Ａ～２Ｅ図は、キャッシュ内の命令の可能性のある位置を示す。

第３図は、本発明による圧縮方策の一部を示す。

第４Ａ～４Ｆ図は、本発明による圧縮された命令の例を図示する。

第５Ａ及び５Ｂ図は、本発明による圧縮された命令フォーマットのテーブルを示す。

第６Ａ図は、命令キャッシュの入力側の機能を示す概念図。

第６Ｂ図は、命令キャッシュの出力側の一部の機能を示す概念図。

第７図は、命令キャッシュ１０４の出力側の機能を示す概念図。

第８図は、本発明によるコードのコンパイル及びリンクを図示する。

第９図は、圧縮及びシャッフルモジュールのフローチャート。

第１０図は、第９図のボックス９０２の詳細。

第１１図は、第１０図のボックス１００５の詳細。

第１２図は、伸張処理を図示する。

発明を実施するための最良の形態

第１Ａ図は、本発明によるプロセッサの概略構成を示している。本発明によるマイクロプロセッサは、ＣＰＵ１０２と、命令キャッシュ１０３と、データキャ

ッシュ１０５とを含んでいる。上記ＣＰＵは広帯域バスにより上記各キャッシュに接続されている。当該マイクロプロセッサは、命令の流れが記憶されるメモリ１０４も含んでいる。

キャッシュ１０３は５１２ビットの倍長ワードを有するように構成されている。ワード中の個々のバイトはアドレス指定可能であるが、ビットはアドレス指定することはできない。バイトは８ビット長である。好ましくは、上記倍長ワードは単一クロックサイクル内で単一ワードとしてアクセス可能なものとする。

前記命令流は、本発明に従い圧縮されたフォーマットの命令として記憶される。圧縮されたフォーマットは、メモリ１０４とキャッシュ１０３の両方で使用される。

第１Ｂ図は、本発明によるＶＬＩＷプロセッサを更に詳細に示している。当該プロセッサはマルチポート・レジスタファイル１５０と、多数の機能ユニット１５１、１５２、１５３、…と、命令イシューレジスタ１５４とを含んでいる。上記マルチポート・レジスタファイルは、機能ユニットからの結果及び機能ユニット用のオペランドを記憶する。前記命令イシューレジスタは、機能ユニット１５１、１５２、１５３、…上で単一クロックサイクル内で並列に開始されるべき各命令部を含むための多数のイシュースロットを有している。以下により詳細に説明する伸張ユニット１５５は、命令キャッシュ１０３からの圧縮された命令をＩＲ１５４により使用可能なフォームに変換する。

圧縮された命令のフォーマット

１．一般的特性

特許請求の範囲に記載された命令フォーマットの好ましい実施例は、５つのイシュースロットを含む命令ワードを持つＶＬＩＷマシンで使用されるように最適化されている。このフォーマットは以下のような特徴を有している。

－　位置合わせされていない可変長命令；

－　命令当たり可変個数の命令部；

－　３つの取り得る大きさの命令部：２６、３４又は４２ビット（２６／３４／４２フォーマットとも呼ぶ）；

－　最も頻繁に使用される３２個の命令部は、他の命令部よりも小型に符号化さ

れる；

－　命令部は保護され又は非保護とされ得る；

－　命令部は零項、単項又は２項のうちの一つである。即ち、命令部は０、１又は２個のオペランドを持つ；

－　命令部は結果無しのものであり得る；

－　命令部は、７又は３２ビットを持つ即値パラメータを含み得る；

－　分岐目標は圧縮されない；

－　命令用のフォーマットビットは先行命令内に位置する。

２．命令の位置合わせ

分岐目標を除いて、各命令はキャッシュ及び主メモリ内ではバイト境界に位置合わせされた形で記憶される。命令は、キャッシュ又は主メモリの何れにおいてもワード又はブロック境界に関しては位置合わせされない。従って、位置合わせされない命令キャッシュアクセスが必要である。

位置合わせされていない命令を取り出すために、プロセッサはキャッシュからクロックサイクル毎に１ワードを取り出す。

以下に述べる圧縮フォーマットから分かるであろうように、分岐目標は単一クロックサイクル内で取り出され得るように、これら分岐目標は圧縮されていない必要があり、且つ、キャッシュの単一ワード内になければならない。分岐目標は、以下の規則に従ってコンパイラ又はプログラマにより位置合わせされる：

ワードの境界が分岐目標内か又は当該分岐目標の終端に正確に位置する場合は、当該分岐目標には次のワードの境界から開始するように詰め込み（padding）が付加される。

好ましいキャッシュは、単一クロックサイクル内で倍長ワードを取り出すので、上記規則はワードの境界を倍長ワードの境界に置き換えるように変更することができる。

通常の位置合わせされていない命令は、順次の命令が現ワードの末尾部と次のワードの開始部とから合成されるようにして取り出される。同様に、全ての後続の命令は２つのキャッシュワードから合成され、各クロックサイクル内では付加ワードが取り出される。

このことは、コードセグメントが再配置される（例えば、リンカ又はローダにおいて）場合は常に、位置合わせが維持されねばならないことを意味する。このことは、コードセグメントのベースアドレスをキャッシュブロック寸法の倍数に再配置することにより達成される。

第２Ａ～２Ｅ図は、本発明によるキャッシュ内の位置決めされない命令の記憶を示している。

第２Ａ図は、本発明による３つの命令ｉ１、ｉ２及びｉ３を伴う２つのキャッシュワードを示している。これらの命令はワードの境界に関しては位置決めされていない。命令ｉ１及びｉ２は完全に一つのキャッシュワード内にあるので、これら命令は分岐目標であり得る。命令ｉ３はワードの境界と交差しているので、分岐目標であってはならない。しかしながら、これらの例示の目的のために、ここではｉ１が（ｉ１のみが）分岐目標であると仮定する。

第２Ｂ図は許容されない状況を示している。分岐目標ｉ１はワードの境界と交差している。従って、コンパイラ又はプログラマは、第２Ｃ図に示すように、命令ｉ１をワードの境界にシフトし、空きの領域を詰め込みバイトで満たさねばならない。

第２Ｄ図は他の許容されない状況を示している。分岐目標命令ｉ１は正確にワードの境界で終了している。この状況でも、第２Ｅ図に示すように、上記命令はワードの境界まで移動され、空き領域は詰め込みで満たされねばならない。

分岐目標は、命令内の命令部というよりも命令でなければならない。以下に述べる命令圧縮技術は通常ノーオペレーション（ダミー命令）を除去する。しかしながら、分岐目標命令は圧縮されていないので、それら命令はプロセッサによって使用されるべきでないイシュースロットを満たすためのノーオペレーションを含んでいなければならない。

３．ビット及びバイト順序

この応用例全体を通して、ビット及びバイト順序はリトル・エンディアン(little endian)である。ビット及びバイトは、最小桁ビットを最初にして以下のように示される。

ビット番号　０…８…１６…

バイト番号　０　１　２

アドレス　　０　１　２

4．命令フォーマット

圧縮された命令は７個までの形式のフィールドを有することができる。これらを以下に掲げる。フォーマットビットのみが必須のフィールドである。

命令はバイト整列された区画からなる。最初の２つのバイトはフォーマットビットと最初の群の２ビット命令部とを含む。全ての他のフィールドは、詰め込みビットを含む第２の２ビット命令部を除いて、バイトの整数倍である。

上述したように、各命令部は２６、３４又は４２ビットを持ち得る。２６ビット命令部はフォーマットビットと共に記憶されるべき２ビット部と、２４ビット部とに分解される。３４ビット命令部は２ビット部と、２４ビット部と、１バイト拡張部とに分解される。４２ビット命令部は２ビット部と、２４ビット部と、２バイト拡張部とに分解される。

A．フォーマットビット

これらは第５節で述べる。５イシュースロットマシンの場合は、１０個のフォーマットビットが必要である。このように、１バイトと２ビットが使用される。

B．２ビット命令部、第１群

各命令部の殆どは以下に説明する２４ビット部、即ち３バイト、に記憶されるが、好ましい命令セットのにとっては、２４ビットでは充分ではない。最も短い命令部でも２６ビットを必要とする。従って、フォーマットビットフィールド用のバイト中に残された前記６ビットが、命令部からの余分なビット、３つの命令部の各々に関し２ビット、を記憶するのに有利に使用することができることが判った。上記２ビット部のために指定された前記６ビットが必要でない場合は、それらビットは詰め込みビットにより満たすことができる。

C．２４ビット命令部、第１群

上記２ビット命令部、第１群に２ビット命令部があったのと同数の２４ビット命令部がある。言い換えると、ここには３個までの３バイト命令部を記憶することができる。

D．２ビット命令部、第２群

３を超えるイシュースロットを持つマシンにおいては、第２群の２ビット及び２４ビット命令部が必要である。第２群の２ビット部は４組の２ビット部を持つバイトからなる。何れかのイシュースロットが使用されない場合は、そのビット部は詰め込みビットで満たされる。詰め込みビットは当該バイトの左側に位置する。５イシュースロットマシンで、全スロットが使用されている場合は、この区画は２群の２ビット部が後続する４個の詰め込みビットを含むであろう。上記５つのイシュースロットは上記２群にわたって分散される：即ち、３個のイシュースロットは第１群に、２個のイシュースロットは第２群となる。

Ｅ．２４ビット命令部、第２群

当該群の２ビット部には対応する群の２４ビット命令部が続く。全スロットが使用される５イシュースロットマシンにおいては、当該群には２つの２４ビット部が在るであろう。

Ｆ．２ビット及び２４ビット部の他の群

非常に幅の広いマシン、即ち６を超えるイシュースロットのマシンにおいては、他の群の２ビット及び２４ビット命令部が必要である。

Ｇ．命令部の拡張

命令の終わりには選択的な８又は１６ビット命令部拡張部のバイト整列された群があり、各命令部拡張部はバイト整列されている。該拡張部は命令部の寸法を、必要なら、基本の２６ビットから３４又は４２ビットに拡張するために用いられる。

命令フォーマットの形式的仕様は：

＜命令＞：：＝

＜命令開始部＞

＜命令中間部＞

＜命令終了部＞

＜命令拡張部＞

＜命令開始部＞：：＝

＜フォーマット：２＊Ｎ＞｛＜詰め込み：１＞｝Ｖ２｛＜２ビット命令部：２｝Ｖ１｛＜２４ビット命令部：２４＞｝Ｖ１

＜命令中間部＞：：＝｛｛＜２ビット命令部：２＞｝４｛＜２４ビット命令部：２４＞｝４｝Ｖ３

＜命令終了部＞：：＝｛＜詰め込み：１＞｝Ｖ５｛＜２ビット命令部：２＞｝Ｖ４｛＜２４ビット命令部：２４＞｝Ｖ４

＜命令拡張部＞：：＝｛＜命令部拡張：０／８／１６＞｝Ｓ

＜詰め込み＞：：＝“０”

ここで、上記において使用される変数は以下のように定義される：

Ｎ＝当該マシンのイシュースロットの数、Ｎ＞１

Ｓ＝当該命令で使用されるイシュースロットの数（０≦Ｓ≦Ｎ）

Ｃ１＝４－（Ｎｍｏｄ４）

（Ｓ≦Ｃ１）の場合は、Ｖ１＝Ｓ及びＶ２＝２＊（Ｃ１－Ｖ１）

（Ｓ＞Ｃ１）の場合は、Ｖ１＝Ｃ１及びＶ２＝０

Ｖ３＝（Ｓ－Ｖ１）ｄｉｖ４

Ｖ４＝（Ｓ－Ｖ１）ｍｏｄ４

（Ｖ４＞０）の場合は、Ｖ５＝２＊（４－Ｖ４）、それ以外の場合は、Ｖ５＝０

表記の説明

：：＝は「と定義される」を意味する。

＜フィールド名：数＞は、コロンの前に示されたフィールドが該コロンの後に示された数のビットを持つことを意味する。

｛＜フィールド名＞｝数は、＜括弧及び｛括弧内に示されたフィールドが｝括弧の後に示された数の回数だけ繰り返されることを意味する。

“０”は、ビット“０”を意味する。

“ｄｉｖ”は、整数除算を意味する。

“ｍｏｄ”は、モジュロを意味する。

：０／８／１６は、当該フィールドが０、８又は１６ビット長であることを意味する。

圧縮された命令の例が第４Ａ～４Ｆ図に示されている。

第４Ａ図は命令部無しの命令を示している。この命令は、フォーマットフィー

ルド用の１０ビットと詰め込みビットのみを持つ６ビットとを含む２バイトを有している。前者は全命令に存在する。後者は通常は２ビット命令部に相当する。ビットフィールドの上部のＸは、当該フィールドが詰め込みを含むことを示している。後の図では、当該フィールドが使用されていることを示すために○が使用される。

第４Ｂ図は１個の２６ビット命令部を伴う命令を示している。当該命令部はバイト３～５に１個の２４ビット部と、バイト２に１個の２ビット部とを含んでいる。使用されている上記２ビットは上部に○の標識が付されている。

第４Ｃ図は２つの２６ビット命令部を持つ命令を示している。第１の２６ビット命令部はバイト３～５に該命令部の２４ビット部を有し、２ビット部フィールドの最後のものに該命令部の超過２ビットを有している。第２の２６ビット命令部はバイト６～８に該命令部の２４ビット部を有し、２ビット部フィールドの最後から２番目のものに該命令部の超過２ビットを有している。

第４Ｄ図は３個の２６ビット命令部を有する命令を示している。２４ビット部はバイト３～１１に位置し、２ビット部はバイト２に上記２４ビット部から逆の順序で位置している。

第４Ｅ図は、４個の命令部を伴う命令を示している。この場合、第２の命令部は２バイトの拡張部を有している。又、第４の命令部は１バイトの拡張部を有している。これら命令部の２４ビット部はバイト３～１１及びバイト１３～１５に記憶されている。最初の３つの命令部の２ビット部はバイト２に位置している。第４の命令部の２ビット部はバイト１２に位置している。第２の命令部の拡張部はバイト１６～１７に位置している。又、第４の命令部の拡張部はバイト１８に位置している。

第４Ｆ図は５つの命令部を伴う命令を示し、これら命令部の各々は１バイトの拡張部を有している。これら拡張部の全ては当該命令の最終部に現れる。

上記例では各拡張部は第２の群の２ビット部の後のみに現れるようになっているが、これら拡張部は同様に３以下の命令部を伴う命令の最終部に現れるようにすることもできる。そのような場合は、第２の群の２ビット部は必要ではないであろう。

命令中の命令部の位置と、これら命令部が送出されるイシュースロットとの間には固定した関係は存在しない。このことは、全てのイシュースロットが使用されてはいないような場合に、命令を短くすることを可能にする。命令部の位置は左から右へと充填される。当該命令のフォーマット部分は、特定の命令部が何のイシュースロットに属するかを示す。例えば、何れかの命令が１個のみの命令部を含む場合は、該命令部は最初の命令部位置に位置し、該命令部はスロット番号１のみに限らず何れのイシュースロットにも送出することができる。伸張ハードウェアは、それらの適したイシュースロットへの経路操作を司る。

コードの１個の系列ブロックを形成する命令の間には、詰め込みバイトは許容されない。詰め込みバイトは、コードの別個のブロックの間で許容される。

５．フォーマットビット

本発明による命令圧縮技術は、当該圧縮された命令により何のイシュースロットが使用されるべきであるかを特定するようなフォーマットフィールドを使用することに特徴がある。取り出し効率を達成するために、フォーマットビットは当該フォーマットビットが関係する命令に先行する命令内に記憶される。このことは、命令取り出しのパイプライン化を可能とする。前記伸張ユニットは、後続の命令に幾つのイシュースロットが在るかにつき、該命令の取り出しに先立って警告される。関連する命令部に先行するフォーマットビットの記憶が、第３図に示されている。圧縮されていない分岐目標である命令１は、命令２において特定される命令部により使用されるイシュースロットを示すフォーマットフィールドを含んでいる。命令２～４は圧縮されている。これら各命令は、後続の命令の命令部により使用されるべきイシュースロットを特定するフォーマットフィールドを含んでいる。

フォーマットビットは以下のように符号化される。Ｎイシュースロット・マシンの場合は２＊Ｎ個のフォーマットビットがある。好ましい実施例の場合は、５個のイシュースロットがある。従って、１０個のフォーマットビットがある。ここでは、フォーマットビットはFormat[j]のようにマトリクス表現で表し、ここでｊはビット番号である。フォーマットビットはＮ群の２ビットに組織化される。ビットFormat[2i]及びFormat[2i+1]はイシュースロットｉに関するフォーマッ

ト情報を与え、ここで0≦i≦Nである。フォーマットビットの意味は以下の表に示される通りである。

表I

Format[2i] lsb	Format[2i+1] msb	意味
0	0	イシュースロットiは使用され、該イシュースロット用の命令部は命令内にある。命令部の大きさは26ビットである。拡張部の大きさは0バイトである。
1	0	イシュースロットiは使用され、該イシュースロット用の命令部は命令内にある。命令部の大きさは34ビットである。拡張部の大きさは1バイトである。
0	1	イシュースロットiは使用され、該イシュースロット用の命令部は命令内にある。命令部の大きさは42ビットである。拡張部の大きさは2バイトである。
1	1	イシュースロットiは使用されず、該イシュースロット用の命令部は命令内には含まれていない。

命令部は左から右への順序でイシュースロットに対応している。例えば、2個のイシュースロットが用いられ、且つ、Format={1,0,1,1,1,1,1,1,0,1,1}の場合は、命令は2個の34ビット命令部を含む。最左端のものはイシュースロット0に経路決めされ、最右端のものはイシュースロット3に経路決めされる。Format={1,1,1,1,1,1,0,1,0,1,0}の場合は、命令は3個の34ビット命令部を含み、最左端の命令部はイシュースロット2に経路決めされ、2番目の命令部はイシュースロット3を意図し、最右端の命令部はイシュースロット4に属する。

分岐目標命令を伸張するのに用いられるフォーマットは、固定数である。好ましい5イシュースロットマシンの場合、固定フォーマット＝{0,1,0,1,0,1,0,1,0

,1}である。

６．命令部のフォーマット

命令部のフォーマットは以下の特性に依存する。

－　零項、単項又は２項のものである；

－　パラメータ的か非パラメータ的か。パラメータ的命令はコード中に即値オペランドを含む。パラメータは異なる大きさのものであり得る。ここでは、param7、即ち７ビットパラメータ及びparam32、即ち３２ビットパラメータがある；

－　結果生成型か、結果無しのものか；

－　長い又は短いオーピーコードである。短いオーピーコードは３２個の最も頻繁なオーピーコードで５ビット長である。長いオーピーコードは８ビット長であり、短いフォーマットで表すことができるのもを含む全オーピーコードを含む。オーピーコード０～３１は、３２個の短いオーピーコード用に確保されている。

－　保護されているか又は保護されていないか。保護されていない命令は真なる保護の一定値を有する。

－　レーテンシィ。フォーマットビットが、命令部が１に等しいレーテンシィか、又は１を越えるレーテンシィを持つかを示す。

－　符号付き／符号無し。フォーマットビットが、パラメータ的命令に関し、当該パラメータが符号付きであるか、符号無しであるかを示す。

圧縮されていない命令フォーマットにおいては、保護されているか又は保護されていないかの特性は定数１の特別レジスタファイルアドレスを用いることにより決定される。保護アドレスフィールドが定数１のアドレスを含む場合は、当該命令部は保護されておらず、それ以外の場合は保護されている。殆どの命令部は保護されたフォーマット又は保護されていないフォーマットの両方で発生し得る。即値命令部、即ち定数をレジスタに転送する命令部は保護フィールドは有さず、常に非保護である。

３２個の短いオーピーコードのリストに何のオーピーコードが含まれるかは、発生頻度の学習に依存し、書かれたソフトウェアの型式に依存して変化し得る。

次の表IIは本発明により使用される命令部フォーマットを掲示している。特に

言及しない限り、全てのフォーマットはパラメータ的でなく、結果を有し、保護されており且つ長いオーピーコードである。表及び図を可能な限り単純化するために、次の表はレーテンシィ及び符号付き／符号無し特性に関する特別なフォームは掲示していない。これらは、フォーマット説明内でＬ及びＳで示されている。非パラメータ的な零項命令部の場合は、単項フォーマットが使用される。その場合は、引数（argument）用のフィールドは定義されていない。

表Ⅱ

命令部型式	大きさ
＜２項－非保護－短い＞	２６
＜単項－param7－非保護－短い＞	２６
＜２項－非保護－param7－結果無し－短い＞	２６
＜単項－短い＞	２６
＜２項－短い＞	３４
＜単項－param7－短い＞	３４
＜２項－param7－結果無し－短い＞	３４
＜２項－非保護＞	３４
＜２項－結果無し＞	３４
＜単項－param7－非保護＞	３４
＜単項＞	３４
＜２項－param7－結果無し＞	４２
＜２項＞	４２
＜単項－param7＞	４２
＜零項－param32＞	４２
＜零項－param32－結果無し＞	４２

全命令部に関し、分岐目標で使用するために４２ビットフォーマットが準備されている。単項及び２項－結果無し命令部に対しては、前記＜２項＞フォーマットを使用することができる。その場合、２項フォーマット中の使用されていないフィールドは未定義の値を有する。短い５ビットのオーピーコードは最上位ビッ

トに０を詰め込むことにより長い８ビットのオーピーコードに変換される。非保護命令部は、保護アドレス値として真なる定数のレジスタファイルアドレスを得る。記憶命令部には、前記４２ビットの＜２項－param7－結果無し＞フォーマットが、通常の３４ビットの＜２項－param7－結果無し－短い＞フォーマットの代わりに使用される（記憶命令部は短い命令部の組に属すると仮定する）。

表IIに現れない命令部型式は、次の別名表に従って、表IIに現れる命令部型式上に分配される。

表II'

フォーマット	別名
零項	単項
単項－結果無し	単項
２項－結果無し－短い	２項－結果無し
零項－param32－短い	零項－param32
零項－param32－結果無し－短い	零項－param32－結果無し
零項－短い	単項
単項－結果無し－短い	単項
２項－結果無し－非保護	２項－結果無し
単項－非保護	単項
２項－param7－結果無し－非保護	２項－param7－結果無し
単項－非保護	単項
２項－param7－結果無し－非保護	２項－param7－結果無し
零項－非保護	単項
単項－結果無し－非保護－短い	２項－非保護－短い
単項－非保護－短い	単項－短い
零項－param32－非保護－短い	零項－param32
零項－param32－結果無し－非保護－短い	零項－param32－結果無し
零項－非保護－短い	単項
単項－結果無し－非保護－短い	単項
単項－長い	２項
２項－長い	２項
２項－結果無し－長い	２項
単項－param7－長い	単項－param7
２項－param7－結果無し－長い	２項－param7－結果なし
零項－param32－長い	零項－param32
零項－param32－結果無し－長い	零項－param32－結果無し
零項－長い	２項
単項－結果無し－長い	２項

次表は命令部に現れるフィールドの表である。

表III

フィールド	大きさ	意味
src1	7	第１オペランドのレジスタファイルアドレス
src2	7	第２オペランドのレジスタファイルアドレス
guard	7	保護のレジスタファイルアドレス
dst	7	結果のレジスタファイルアドレス
param	7／32	７ビットパラメータ又は３２ビット即値
op code	5／8	５ビットの短いオーピーコード又は８ビットの長いオーピーコード

第５図は命令部の符号化の全仕様を含んでいる。

７．命令フォーマットの拡張

命令フォーマット内には、４２ビットの最大寸法内での符号化が可能な限りにおいて、新たな命令部及び命令部フォームを追加するための幾らかの柔軟性がある。

フォーマットは７ビットのレジスタファイルアドレスに基づく。異なる大きさのレジスタファイルアドレスに関しては、フォーマット及び伸張ハードウェアの再設計が必要である。

当該フォーマットは可変数のイシュースロットを持つマシン上で使用することができる。しかしながら、命令の最大寸法は命令キャッシュ内のワード寸法により規制される。４イシュースロットマシンにおいては、最大命令寸法は４個の４２ビット命令部＋８個のフォーマットビットを用いて２２バイト（１７６ビット）となる。５イシュースロットマシンにおいては、最大命令寸法は５個の４２

ビット命令部＋１０個のフォーマットビットを用いて２８バイト（２２４ビット）となる。

６イシュースロットマシンにおいては、最大命令寸法は６個の４２ビット命令部＋１２個のフォーマットビットを用いて２６４ビットとなる。ワードの大きさが２５６ビットに制限され、且つ、６個のイシュースロットが所望の場合は、スケジューラは１命令中で４２ビットフォーマットの最大で５個の命令部しか使用しないよう制限される。分岐目標用の固定のフォーマットは、４２ビットの５個のイシュースロットと３４ビットの１個のイシュースロットを使用しなければならない。

命令の圧縮

第８図は、どのようにしてソースコードがロード可能な圧縮されたオブジェクトモジュールになるかを示す図である。先ず、ソースコード８０１はコンパイラ８０２によりコンパイルされ、オブジェクトモジュールの最初の組８０３が生成される。これらモジュールはリンカ８０４により結合されて、第２の形式のオブジェクトモジュール８０５が生成される。このモジュールは、次いで、８０６で圧縮及びシャッフルされ、これによりロード可能なモジュール８０７が生成される。

どのような標準コンパイラ及びリンカでも使用することができる。オブジェクトモジュールIIは多数の標準データ構造を含んでいる。これらは、ヘッダ；グローバル及びローカルシンボルテーブル；再配置情報用のリファレンステーブル；セクションテーブル；及びデバッグ情報を含み、これらの内の幾つかは圧縮及びシャッフルモジュール８０７により使用される。オブジェクトモジュールIIは、処理すべき命令が居るテキスト区画と、当該テキストが何のソースファイルから来たかを追跡するソース区画とを含む区画も有している。

上記圧縮及びシャッフルモジュールの高レベルフローチャートが第９図に示されている。この場合、９０１においてモジュールIIが読み込まれる。９０２においては前記テキスト区画が処理される。９０３においては他の区画が処理される。９０４においてはヘッダが更新される。９０５においては、オブジェクトモジュ

ールが出力される。

第１０図は、上記ボックス９０２を展開したものである。この場合、1001では参照テーブル、即ち再配置情報が収集される。1002では分岐目標が収集される。何故なら、これらは圧縮されないからである。1003では、当該ソフトウェアは、ソース区画中にもっとファイルがあるかチェックする。もしあるなら、1004で次のファイルに対応する部分が取り出される。次いで、1005で当該部分が圧縮される。1006では、上記ソース区画のファイル情報が更新される。1007では、ローカルシンボルテーブルが更新される。

上記ソース区画に一旦ファイルが無くなると、上記グローバルシンボルテーブルが1008において更新される。次いで、1009では、テキスト区画のアドレスリファレンスが更新される。次に、1010では、２５６ビットのシャッフルが実行される。このようなシャッフルの動機付けを以下に説明する。

第１１図は、上記ボックス1005を展開したものである。最初に、1101において圧縮されるべき命令がもっとあるか否かが判断される。もしあるなら、次の命令が1102で取り出される。次いで、当該命令の各命令部が1103において第５Ａ及び５Ｂ図のテーブルに従って圧縮され、分散テーブルが1108において更新される。この分散テーブルは圧縮及びシャッフルの結果として必要となる新たなデータ構造であり、以下に更に説明する。次いで、1104において、一つの命令における全命令部及び次の命令のフォーマットビットが第４Ａ～４Ｅ図のように合成される。現命令がアドレスを含む場合は、次いで、1105において前記参照テーブルの再配置情報が更新されなければならない。1106では、次のテキスト区画のアドレスリファレンスを更新するに要する情報が収集される。1007では、圧縮された命令が出力ビット系列の最後に付加され、制御がボックス1101に戻される。最早命令がない場合は、制御はボックス1006に戻される。

圧縮を扱う機能は以下に掲示するように種々のモジュールにおいて実施することができる。

表IV

モジュール名	実施される機能の識別
scheme-table	第５Ａ及び５Ｂ図の表の読取可能版
comp-shuffle.c	２５６ビットシャッフル、ボックス1010参照
comp-scheme.c	ボックス1103～1104
comp-bitstring.c	ボックス1005及び1009
comp-main.c	第９及び１０図の主フローを制御
comp-src,c, comp-reference.c, comp-misc.c, comp-btarget.c	第１１図に掲げた他の機能を実施するための雑サポートルーチン

本発明の圧縮及びシャッフルの結果として必要になる分散テーブルは、以下のように説明することができる。

前記参照テーブルは、当該命令流により使用されるアドレスの位置のリストと、これらの位置に掲示された実際のアドレスの対応するリストとを含んでいる。コードが圧縮され、且つ、ロードされる場合は、これらのアドレスは更新されねばならない。従って、上記参照テーブルは、これらの時点で上記更新を可能にするために使用される。

しかしながら、コードが圧縮され且つシャッフルされる場合は、各アドレスの実際のビットは互いに分離され且つ再配列される。従って、前記分散テーブルは、参照テーブルにおける各アドレスに対して、各ビットが何処に位置されるかを掲示する。好ましい実施例では、当該テーブルはビットフィールドの幅、ソーステキスト中の当該アドレスの対応するインデクスからのオフセット、デスティネーションテキスト内のアドレスの対応するインデクスからの対応するオフセットを掲示する。

プロセッサで実行するためにオブジェクトモジュールIIIがロードされると、前記分散テーブルは、ビットがシャッフル解除される前でも参照テーブルに掲示されたアドレスが更新されることを可能にする。

上記分散テーブルは、例えば、一組の分散記述子を含む。各分散記述子は一群の三つ組み（デスティネーションオフセット、幅、ソースオフセット）と、当該記述子内の三つ組みの数を示す符号無し整数とを含んでいる。

例えば、分散記述子が３個の三つ組み（０、７、３）、（７、４、１５）、（１１、５、２３）を有していると仮定する。又、ソースフィールドがテキスト区画のビットフィールドの位置３２０にあると仮定する。実際のアドレスフィールドのビットを得るために、以下のようにする。アドレスフィールドのビット０～６（７ビット）はビット系列からの位置３２３（３２０－３）～３２９からとする。アドレスフィールドのビット７～１０はビット系列のビット３３５～３３８からとする。又、アドレスフィールドのビット１１～１５はビット系列のビット３４３～３４７からとする。このように、アドレスフィールドは長さ１６を有する。

オブジェクトモジュールにおいては、参照記述子のリストがビット系列と関連付けられる。各参照記述子はビット系列中の一つのビットフィールドを参照する。各参照記述子は、分散テーブルへの、当該ビットフィールドのビットが前記ビット系列内で分散された態様についての情報を持つ分散記述子を見つけることができるインデクスを含んでいる。例えば、ビットフィールドがビット系列内の位置１１を有し、当該ビットフィールドに対応する分散記述子が単一のエントリ（０、１８、０）を有する場合は、実際のソースオフセットは、上記位置とソースオフセットとを一緒に加算する（１１＋０）ことにより得られる。

命令の伸張

ＶＬＩＷプロセッサが上述したようにして圧縮された命令を処理するには、これら命令は伸張されねばならない。伸張の後、これら命令はＮ個のイシュースロットを持つ命令レジスタを満たし、ここでＮは好ましい実施例の場合は５である。第１２図は上記伸張の過程を示す図である。命令はメモリ1201、即ち主メモリ１０４か又は命令キャッシュ１０５の何れか、から到来する。次いで、伸張1203される前に、以下に述べるように、シャッフル解除1201されねばならない。伸張1203の後、これら命令はＣＰＵ1204に進むことができる。

伸張された各命令部は２個のフォーマットビットと４２ビットの命令部とを有

している。上記２個のフォーマットビットは、４つの可能性のある命令部の長さ（イシュースロット不使用、２６ビット、３４ビット又は４２ビット）の一つを指す。これらのフォーマットビットは、上記第５節における「Format」と同一の値を有する。命令部が２６ビット又は３４ビットの大きさを有する場合は、上位の８又は１６ビットは未定義である。イシュースロットが不使用の場合は、フォーマットビットにより示されるように、全命令部ビットは未定義であり、ＣＰＵは当該オーピーコードをノーオペレーションコードに置換しなければならない（さもなくば、機能ユニットにノーオペレーションを指示しなければならない）。

正式には、伸張された命令のフォーマットは、

＜伸張された命令＞：：＝｛＜伸張された命令部＞｝Ｎ

＜伸張された命令部＞：：＝＜命令部：４２＞＜フォーマット：２＞

命令部は（前記）表IIIのようなフォーマットを有している。

付録Ａは、伸張ユニットの機能を規定するVERILOGコードである。VERILOGコードは、カリフォルニア州サンジョゼのカデンス・デザイン・システムズ・インクにより製造されるVERILOGシミュレータへの入力として使用される標準フォーマットである。上記コードはカリフォルニア州マウンテンビューのシノプシスにより製造されるデザインコンパイラに直接入力することもでき、これにより当該コードを伸張する伸張ユニットの回路図を作成することができる。VERILOGコードは当該伸張ユニットのピンのリストを規定し、これらは：

表Ｖ

群になったピンの数	ピン群の名称	ピン群の説明
５１２	data512	メモリ、即ち命令キャッシュから又は主メモリ、からの５１２ビットの入力データワード
３２	PC	プログラムカウンタの入力
４４	operation4	イシュースロット４の内容の出力
４４	operation3	イシュースロット３の内容の出力
４４	operation2	イシュースロット２の内容の出力
４４	operation1	イシュースロット１の内容の出力
４４	operation0	イシュースロット０の内容の出力
１０	format-out	命令部内のフォーマットビットの複製の出力
３２	first-word	プログラムカウンタにより指された最初の３２ビットの出力
1	format-ctrl0	分岐目標か否か？
各々１	reissue1 stall-in freeze reset clk	グローバルパイプライン制御信号の入力

Data512は倍長ワードであり、現在関心のある命令を含んでいる。上記において、プログラムカウンタＰＣは下記のアルゴリズムに従いData512を決定するために使用される：

Ａ：＝｛ＰＣ［３１：８］，８’ｂ０｝

もしＰＣ［５］＝０なら、

Data512’：＝｛Ｍ（Ａ），Ｍ（Ａ＋３２）｝

その他の場合は、Data512’：＝｛Ｍ（Ａ＋３２），Ｍ（Ａ）｝

ここで、Ａはメモリ中の関心のある命令を含む単長ワードのアドレスである；

又、８’ｂ０は零化された８ビットを意味し；

Ｍ（Ａ）はＡによりアドレス指定されたメモリのワード；

Ｍ（Ａ＋３２）はＡ＋３２によりアドレス指定されたメモリのワード；

Data512'はdata512のシャッフルされものである。

このことは、奇数ワードがアドレス指定された場合はワードがスワップされることを意味する。

命令部は上記伸張ユニットにより部分的に伸張された形態で送出される。何故なら、当該命令部フィールドは常に同一ビット位置にあるとは限らないからである。それらビット位置から上記命令部フィールドを抽出するために何らかの他の処理がなされねばならず、それらの殆どはＣＰＵパイプラインの命令デコード段中で好適になされる。これを、各命令部フィールドに関して以下に説明する。

src1

src1フィールドは固定位置にあり、アドレスとしてレジスタファイルに直接渡すことができる。３２ビット即値命令部のみがsrc1フィールドを使用しない。この場合は、ＣＰＵ制御は上記レジスタファイルからのsrc1オペランドは使用しない。

src2

src2フィールドは使用される場合は固定の位置にあり、アドレスとしてレジスタファイルに直接渡すことができる。使用されない場合は、src2フィールドは不定値を持つ。ＣＰＵ制御は、レジスタファイルから読み取られた「ダミー」src2値は使用されないようにする。

guard

保護フィールドは使用される場合は固定の位置にあり、アドレスとしてレジスタファイルに直接渡すことができる。レジスタファイルへのアクセスと同時に、ＣＰＵ制御は当該命令部のオーピーコードとフォーマットビットを検査する。当該命令部が保護されていない場合は、ＲＦ（レジスタファイル）から読みとられた保護値は定数ＴＲＵＥにより置換される。

オーピーコード

短及び長オーピーコードは当該命令部中の固定の位置に得られる。それらは、ビット位置２１～３０＋２フォーマットビットにある。これらは、デコード用の最大時間内でオーピーコードデコードに直接供給することができる。

dst

dstフィールドはレイテンシ０の３２ビット即値命令部の場合は非常に即座に必要となる。この特別な場合は、ビット３３及びフォーマットビットを検査することによりＣＰＵにより即座に検出される。他の全ての場合では、当該命令部中の何処にdstフィールドがあるか（多数の位置にあり得る）デコードし、それを取り出すために、命令デコードパイプライン状態中では全クロックサイクルを利用することができる。

３２ビット即値

もし３２ビットの即値がある場合は、それは当該命令部の固定位置にある。７個の最下位ビットは、７ビットパラメータのように、同一位置のsrc2フィールドにある。

７ビットパラメータ

７ビットパラメータがある場合は、それは当該命令部のsrc2フィールドにある。一つ例外がある：オフセット命令部を伴うストアである。この命令部の場合、７ビットパラメータは種々の位置にあり得、データキャッシュへの特別な７ビット即値バス上に多重化される。

ビットスウィズリング

命令が長い場合、例えば５１２ビット倍長ワードの場合、キャッシュ構造は複雑になる。命令のビットをスウィズルして、チップの配置を単純化するのが有利である。ここでは、スウィズル及びシャッフルなる語は、同じ事を意味するように使用されている。以下は、ビットをスウィズルするためのアルゴリズムである。

```
for (k=0; k<4; k=k+1)
        for (i=0; j<8; i=i+1)
                for (j=0; j<8; j=j+1)
                begin
                        word_shuffled[k*64+j*8+i]=
                        -word_unshuffled[(4*i+j)*8+j]
        end
```

ここで、ｉ、ｊ及びｋは整数インデクスであり；word_shuffledはシャッフルされたワードのビットを記憶するマトリクスであり；word_unshuffledはシャッフル解除されたワードのビットを記憶するマトリクスである。

キャッシュ構造

第６Ａ図は、ＶＬＩＷ命令の効率的な処理に有効なキャッシュ構造の入力に対する動作を示している。このキャッシュは、各々が２ｋバイトの１６個のバンク６０１～６１６を含んでいる。これらのバンクは入力バス６１７を分け合っている。これらキャッシュは２つのスタックに分割されている。左側のスタックは「ロー」と呼び、右のスタックを「ハイ」と呼ぶ。

当該キャッシュは一度には一方のバンクに、従って一度には４バイトのみを、入力することができる。アドレス指定が、どのバンクのどの４バイトが満たされるかを決定する。キャッシュに記憶されるべき５１２ビットワードの各々に対して、各バンクには４バイトが記憶される。各バンクの陰影を付けた部分は、各バンクの対応する部分が所与のワードをロードするためのものであることを示している。これらの陰影を付けた部分は図示のためのみのものである。如何なるワードも、これらバンクの如何なる組の対応する部分にロードすることもできる。

前述したアルゴリズムによるスウィズリングの後、スウィズルされたワードの連続した４つのバイト部分が、６０８、６１６、６０６、６１４、６０４、６１２、６０２、６１０、６０７、６１５、６０５、６１３、６０３、６１１、６０１、６０９の順で各バンクにロードされる。スウィズルされたワードの上記４バイト部分をロードする順序は、これらバンクを表すボックス内にローマ数字で示されている。

第６Ｂ図は、スウィズルされたワードが当該キャッシュからどの様に読み出されるかを示している。第６Ｂ図はロー側スタックのバンクの陰影部分のみを示している。ハイ側部分も同様である。陰影部分６０１ａ～６０８ａは各々３２ビットを有している。各ビットは図示の接続を用いてbus256lowと呼ばれる出力バス上にロードされる。即ち、608a-bit0, 607a-bit0, …, 601a-bit0; 608a-bit1, 607a-bit1, …, 601a-bit1; …; 608a-bit31, 607a-bit31, …, 601a-bit31の順である。これらの接続を用いて、ワードは自動的に適切なビット順にスウィズル解除される。

配線６２０、６２１、…、６２２の各束は、一緒になって出力端bus256lowを形成する。これら配線は当該キャッシュを上記出力端に向かって交差することなく通過する。

出力に関しては当該キャッシュは第７図のようである。各ビットは制御ユニット７０４の制御の下で、シフト回路網７０３を介してスタック・ロー７０１とスタック・ハイ７０２から読み出され、上記回路網は上記各ビットが前述した出力順序になることを保証する。このようにして、５１２ビットワードの全出力が束線６２０、６２１、…、６２２及び同様の配線が交差することなく達成される。

上記においては、圧縮された命令を使用するＶＬＩＷプロセッサを説明した。このＶＬＩＷプロセッサは複数のイシュースロットを持つ命令イシューレジスタを有し、各イシュースロットは各命令部を記憶するものであり、全ての命令部は同一クロックサイクル内で実行を開始する。当該ＶＬＩＷプロセッサは命令レジスタに記憶された各命令部を実行するための複数の機能ユニットを有している。又、当該ＶＬＩＷプロセッサは伸張された命令を上記命令イシューレジスタへ供給する伸張ユニットを有し、該伸張ユニットは圧縮命令記憶媒体から圧縮された命令を受け取り、これら圧縮された命令を伸張する。前記圧縮された命令のうちの少なくとも１個は少なくとも１個の命令部を有し、各命令部は当該命令部に圧縮命令部長を割り当てる圧縮方策に従って圧縮される。圧縮された命令部の長さは複数の有限の長さから選択され、これら有限の長さは少なくとも２個の零でない長を含み、これら有限の長さの何れが選択されるかは当該命令部の少なくとも

一つの特徴に依存する。

好ましくは、上記の命令部長さの組み合わせは｛０、２６、３４、４２｝である。又、好ましくは上記少なくとも一つの特徴は下記のものの少なくとも一つである：

－　短縮されたオーピーコード；

－　保護されているか又は保護されていないか；

－　結果無し；

－　固定ビット数の即値パラメータ；及び

－　零項、単項又は２項の。

上記固定数は、好ましくは７又は３２の何れかである。好ましくは、当該プロセッサは複数の上記のような命令を有し、これらのうち１つの命令は分岐目標であり、この一つの命令は圧縮されない。好ましくは、各命令内の各命令部フィールドは次のものの少なくとも一つを特定する副フィールドを含む：第１オペランドのレジスタファイルアドレス；第２オペランドのレジスタファイルアドレス；保護情報のレジスタファイルアドレス；結果のレジスタファイルアドレス；即値パラメータ；及びオーピーコード。好ましくは、各命令は複数の各フォーマットを特定するフォーマットフィールドを有し、ここで各フォーマットは後続の命令の各命令部のためのものである。好ましくは、上記の圧縮されたフォーマットは、何れかの命令により使用されるべきＶＬＩＷプロセッサのイシュースロットを特定するフォーマットフィールドを有する。

好ましくは、少なくとも一つのフィールドが当該命令部を特定する。当該命令部を特定するフィールドは、少なくとも１個のバイト位置合わせされた副フィールドを有する。好ましくは、少なくとも一つの命令部副フィールドは、前記フォーマットフィールドと同一のバイト内に位置する。このように、各命令はワード境界ではなく、バイト境界に位置合わせされ得る。好ましくは、フォーマットフィールドは、或る閾量を超えるイシュースロットが使用されるべきであることを特定し、且つ、更に上記フォーマットフィールドと同一のバイト内に位置する少なくとも一つの第１命令部副フィールド、命令部を特定する複数の副フィールド

及び他の副フィールドとは別のバイト中に位置する少なくとも一つの第２命令部副フィールドを有する。

付録 A

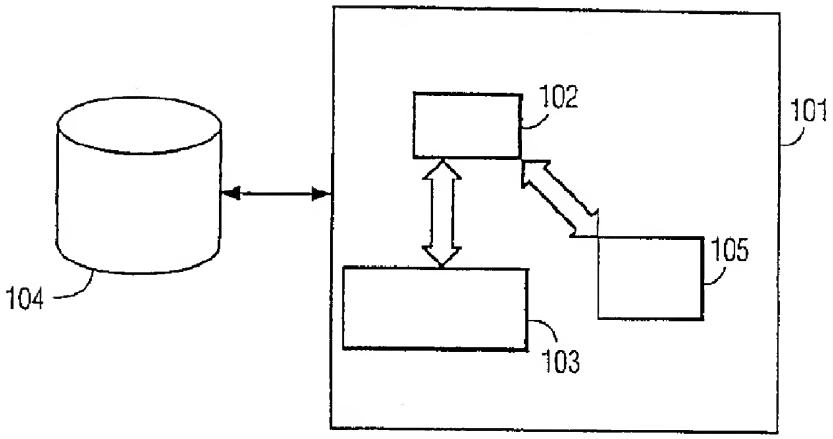
```
// Verilog HDL for Icache, ic_decompression _behavioral

`define FIXED_FORMAT 10'b1010101010
`define NOP_OPERATION 42'b0

module    ic_decompression (data512, pc,
                        operation4, operation3, operation2, operation1,
                              operation0, format_out, first_word,
                        format_ctrl0,
                        reissue1, stall_in, freeze, reset, clk);

input [511:0] data512;
input [31:0] pc;
output [43:0] operation0, operation1, operation2, operation3, operation4;
reg [43:0] operation0, operation1, operation2, operation3, operation4;
output [31:0] first_word;
reg [31:0] first_word;
input format_ctrl0;
input reissue1, freeze, reset, clk;
wire [9:0] format_out1;
output [9:0] format_out;
reg [9:0] format_outA, format_p;
input stall_in;

// local
reg [9:0] format_out0;
reg [31:0] pc_p;
reg format_ctrl;

reg [9:0] format;
reg used0, used1, used2, used3, used4;
reg [1:0] size0, size1, size2, size3, size4;

reg [511:0] data512shift;
reg [255:0] data256;
reg [2:0] pos1,  pos2,  pos3,  pos4, pos_ext;

reg [25:0] fix0, fix1, fix2, fix3, fix4;
reg [79:0] extension0, extension1, extension2, extension3, extension4;
reg [15:0] ext0, ext1, ext2, ext3, ext4;
reg reset_p;

// format pipe
always @(posedge clk)
begin
reset_p <= reset;

if (reset_p)
begin
      // force NOP operations on instructions
      operation0[42] <= `ONE;
      operation0[43] <= `ONE;
      operation1[42] <= `ONE;
      operation1[43] <= `ONE;
      operation2[42] <= `ONE;
      operation2[43] <= `ONE;
      operation3[42] <= `ONE;
      operation3[43] <= `ONE;
      operation4[42] <= `ONE;
      operation4[43] <= `ONE;
end
else if (~stall_in)
begin

pc_p <= pc;
format_ctrl <= format_ctrl0;
```

```
if (format_ctrl)
   format = `FIXED_FORMAT;
else
   format = format_out;

used0 = ~(format[1] & format[0]);
size0 = used0 ? format[1:0] : 2'b0;

used1 = ~(format[3] & format[2]);
size1 = used1 ? format[3:2] : 2'b0;

used2 = ~(format[5] & format[4]);
size2 = used2 ? format[5:4] : 2'b0;

used3 = ~(format[7] & format[6]);
size3 = used3 ? format[7:6] : 2'b0;

used4 = ~(format[9] & format[8]);
size4 = used4 ? format[9:8] : 2'b0;

// first alignment stage
// rotate the 512 bit word right over a distance between 0 and 64 byte
//
// the rotate is implemented here by swapping the left and right word if the
// distance is more than 32 byte and then perform a right shift over a distance
between
// 0 and 32 byte.

data512shift = pc_p[5] ?
           {data512[255:0], data512[511:256]} >>  {pc_p[4:0], 3'b0} :
           data512 >> {pc_p[4:0], 3'b0};

data256 = data512shift[255:0];

// extract format bits
format_out0 = data256[9:0];

// access first word
first_word <= data256[31:0];

// Notes: - the value for pos_ext==0 is don't care
//        - for values pos_ext < 5 , less than 80 bits are needed

// determine the position of issue slots

//pos0 = 0;
pos1 = used0;
pos2 = used0 + used1;
pos3 = used0 + used1 + used2;
pos4 = used0 + used1 + used2 + used3;

// mux the fixed part of issue slots, combine the 24-bit part and the 2-bit part

fix0 = used0 ? {data256[15:14], data256[(0+1)*24+15 : 0*24+16]} : `NOP_OPERATION
;

fix1 = used1 ?
                (
                  pos1 == 0 ? {data256[15:14], data256[(0+1)*24+15 : 0*24+16]} :
                              {data256[13:12], data256[(1+1)*24+15 : 1*24+16]}
                )
              : `NOP_OPERATION;

fix2 = used2 ?
                (
```

```
                    pos2 == 0 ? {data256[15:14], data256[(0+1)*24+15 : 0*24+16]} :
                    pos2 == 1 ? {data256[13:12], data256[(1+1)*24+15 : 1*24+16]} :
                                {data256[11:10], data256[(2+1)*24+15 : 2*24+16]}
                  }
                : 'NOP_OPERATION;

fix3 = used3 ?
                  {
                    pos3 == 0 ? {data256[15:14], data256[(0+1)*24+15 : 0*24+16]} :
                    pos3 == 1 ? {data256[13:12], data256[(1+1)*24+15 : 1*24+16]} :
                    pos3 == 2 ? {data256[11:10], data256[(2+1)*24+15 : 2*24+16]} :
                                {data256[95:94], data256[(0+1)*24+95 : 0*24+96]}
                  }
                : 'NOP_OPERATION;

fix4 = used4 ?
                  {
                    pos4 == 0 ? {data256[15:14], data256[(0+1)*24+15 : 0*24+16]} :
                    pos4 == 1 ? {data256[13:12], data256[(1+1)*24+15 : 1*24+16]} :
                    pos4 == 2 ? {data256[11:10], data256[(2+1)*24+15 : 2*24+16]} :
                    pos4 == 3 ? {data256[95:94], data256[(0+1)*24+95 : 0*24+96]} :
                                {data256[93:92], data256[(1+1)*24+95 : 1*24+96]}
                  }
                : 'NOP_OPERATION;

// determine the position of the extension part
pos_ext = used0 + used1 + used2 + used3 + used4;

// determine the extension
extension0 =
                    pos_ext == 0 ? data256[0*24+80-1+16 : 0*24+16] :
                    pos_ext == 1 ? data256[1*24+80-1+16 : 1*24+16] :
                    pos_ext == 2 ? data256[2*24+80-1+16 : 2*24+16] :
                    pos_ext == 3 ? data256[3*24+80-1+16 : 3*24+16] :
                    pos_ext == 4 ? data256[1*24+80-1+96 : 1*24+96] :
                                   data256[2*24+80-1+96 : 2*24+96];

// shift the Extension part
extension1 = extension0 >> (size0 , 3'b0};
extension2 = extension1 >> (size1 , 3'b0};
extension3 = extension2 >> (size2 , 3'b0};
extension4 = extension3 >> (size3 , 3'b0};

ext0 = extension0[15:0];
ext1 = extension1[15:0];
ext2 = extension2[15:0];
ext3 = extension3[15:0];
ext4 = extension4[15:0];

// assemble instruction
//operation0 <= {format_out0[1:0], ext0, fix0};
//operation1 <= {format_out0[3:2], ext1, fix1};
//operation2 <= {format_out0[5:4], ext2, fix2};
//operation3 <= {format_out0[7:6], ext3, fix3};
//operation4 <= {format_out0[9:8], ext4, fix4};
operation0 <= {format[1:0], ext0, fix0};
operation1 <= {format[3:2], ext1, fix1};
operation2 <= {format[5:4], ext2, fix2};
operation3 <= {format[7:6], ext3, fix3};
operation4 <= {format[9:8], ext4, fix4};

if (~freeze | reissue1)
begin
   format_outA <= format_out0;
end
```

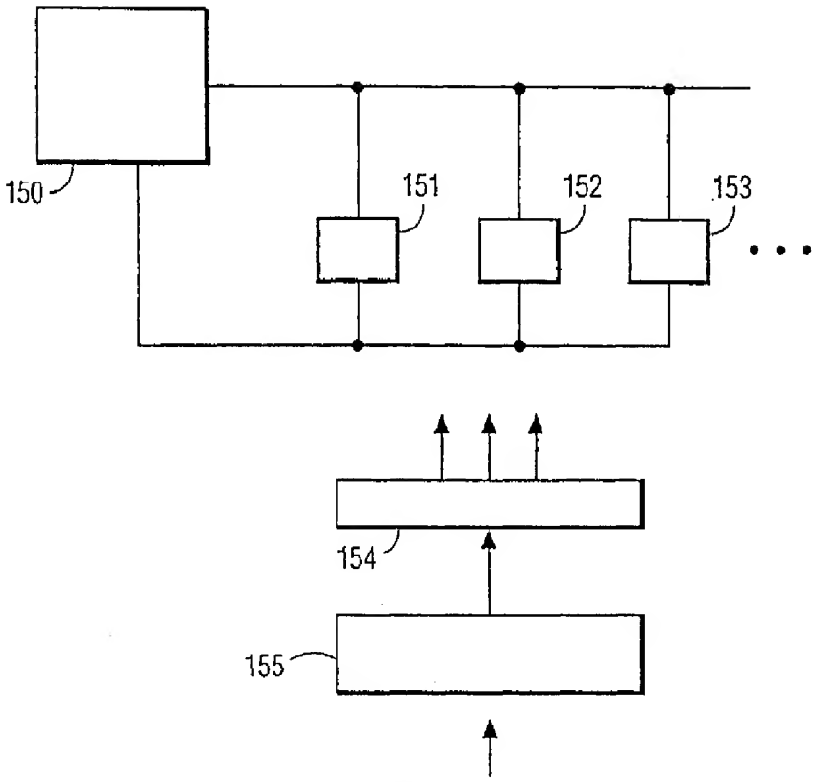
```
if (~freeze)
begin
  format_p <= format_outA;
end


end
end

assign format_out = reissue1 ? format_p : format_outA;
assign format_out1 = format;


endmodule
```

【図1】



第1A図

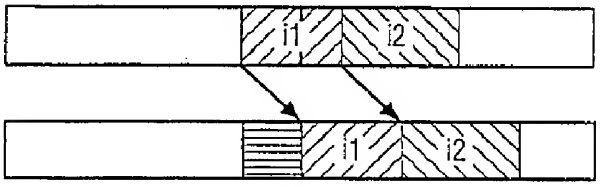
【図１Ｂ】



第1B図

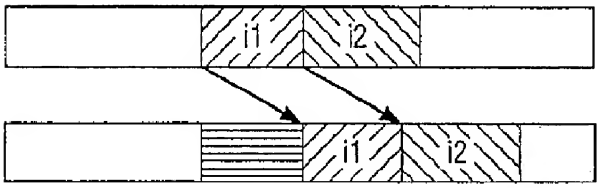
【図２】

第2A図



第2B図

第2C図

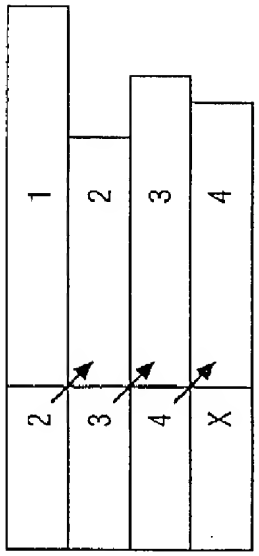


第2D図

第2E図

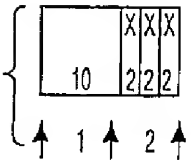
【図３】

第３図

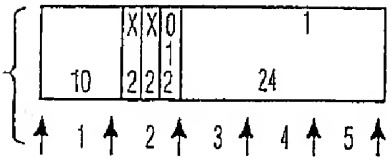


【図４】

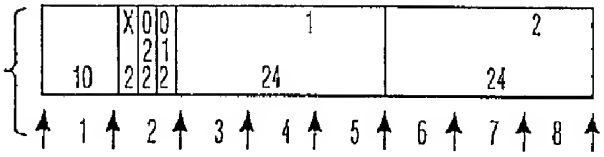
第４Ａ図



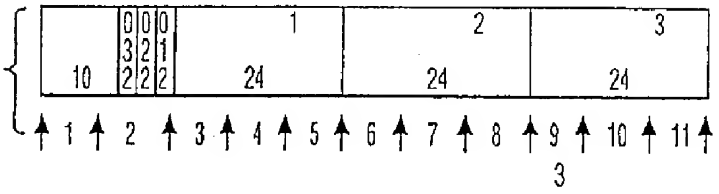
第４Ｂ図



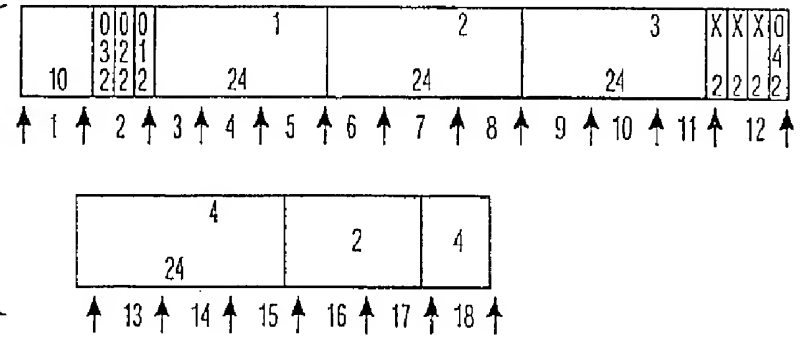
第４Ｃ図



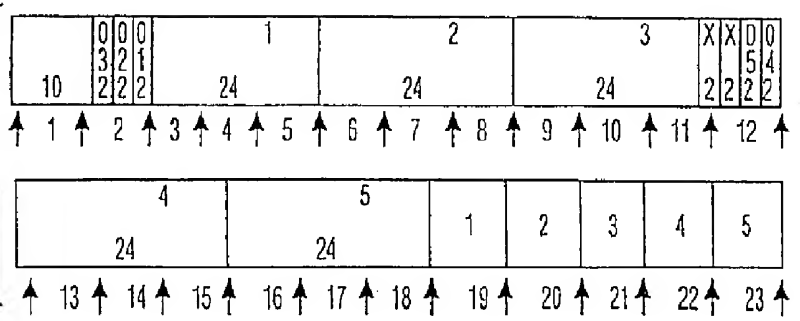
第４Ｄ図



第４Ｅ図



第４Ｆ図



	24-bit				2-bit		
	0-6	7-13	14-20	21-23	24-25	26-34-41	
26-							
<binary-unguarded-short>	src1[0:6]	src2[0:6]	dst[0:6]	opcode[0:2]	opcode[3:4]		26
<unary-param7-unguarded-short>	src1[0:6]	param[0:6]	dst[0:6]	opcode[0:2]	opcode[3:4]		26
<binary-unguarded-param7-resultless-short>	src1[0:6]	src2[0:6]	param[0:6]	opcode[0:2]	opcode[3:4]		26
<unary-short>	src1[0:6]	dst[0:6]	guard[0:5]	opcode[0:2]	opcode[3:4]		26
34-							
<binary-short> iadd,etc.	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	dst[0:6] 0	34
<unary-param-7-short>	src1[0:6]	param[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	dst[0:6] 0	34
<binary-param7-resultless-short>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	param [0:6] 0	34
<binary-unguarded>	src1[0:6]	src2[0:6]	dst[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]XL011	34
<binary-resultless>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]X1001	34
<unary-param7-un-guarded>	src1[0:6]	param[0:6]	dst[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]SL111	34
<unary>	src1[0:6]	dst[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]XL101	34

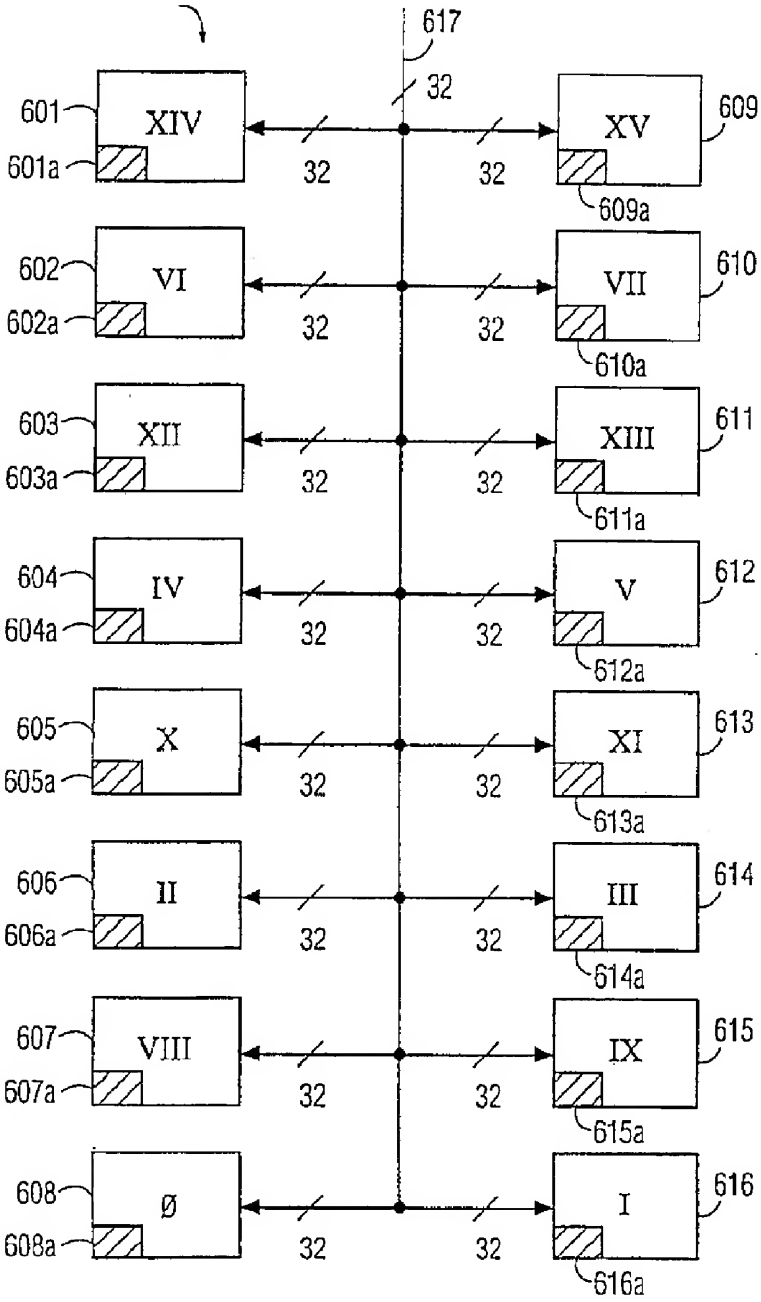
第5A図

【図５】

42-								
<binary-param7-resultless>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]SXX100param[0:6]	42
<binary>	src1[0:6]	src2[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]XL0101 dst[0:6]	42
<unary-param7>	src1[0:6]	param[0:6]	guard[0:6]	opcode[0:2]	opcode[3:4]	opcode[5:7]SL1101 dst[0:6]	42
<zeroary-param32>	param[7:13]	param[0:6]	dst[0:6]	param[14:16]	param[17:18]	param[19:23]XX1 param[24:31]	42
<zeroary-param32-resultless>	param[7:13]	param[0:6]	guard[0:6]	param[14:16]	param[17:18]	param[19:23]000 param[24:31]	42
<zeroary-param32-resultless>	param[7:13]	param[0:6]	guard[0:6]	param[14:16]	param[17:18]	param[19:23]100 param[24:31]	42

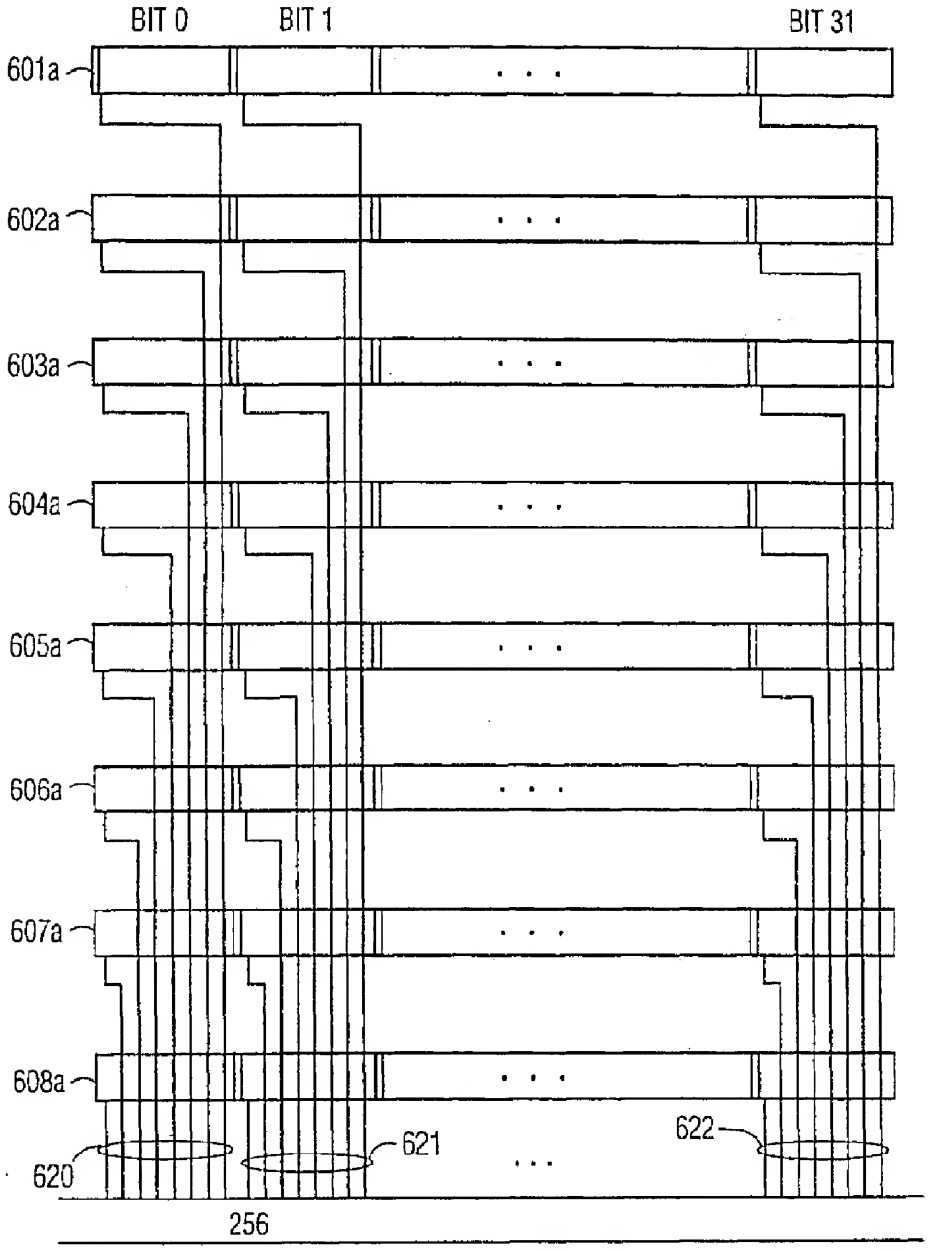
第5B図

【図６】


617
32
601
XIV
601a
32
32
XV
609
609a
602
VI
602a
32
32
VII
610
610a
603
XII
603a
32
32
XIII
611
611a
604
IV
604a
32
32
V
612
612a
605
X
605a
32
32
XI
613
613a
606
II
606a
32
32
III
614
614a
607
VIII
607a
32
32
IX
615
615a
608
Ø
608a
32
32
I
616
616a

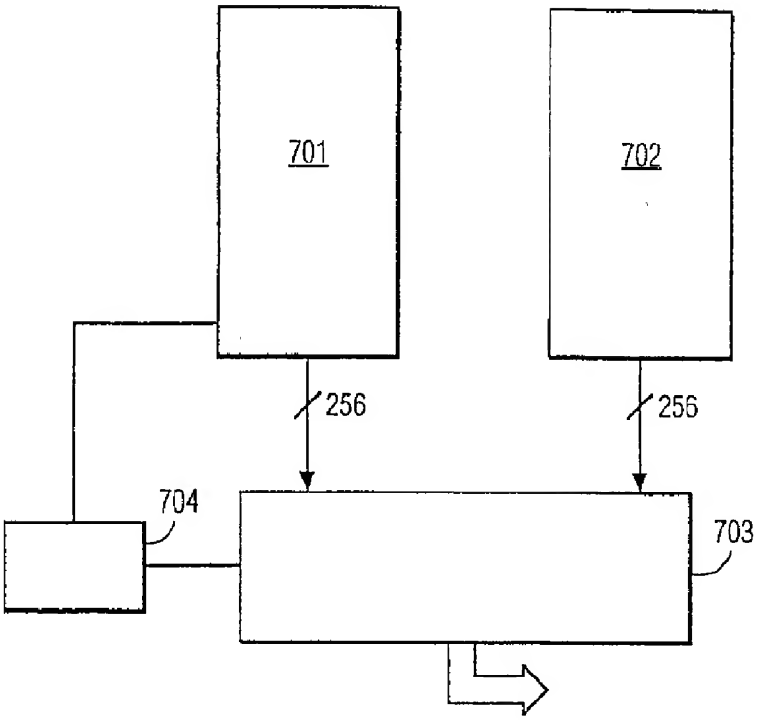
第6A図

【図６】


BIT 0
BIT 1
BIT 31
601a
602a
603a
604a
605a
606a
607a
608a
620
621
622
256

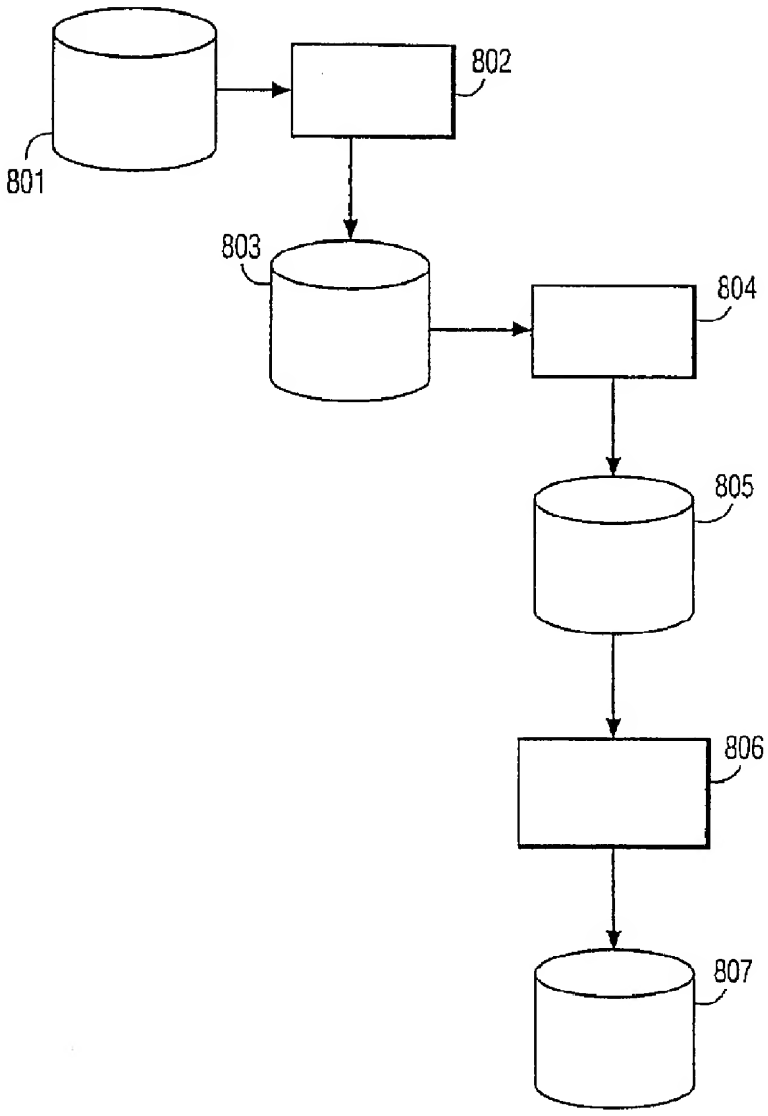
第6B図

【図７】



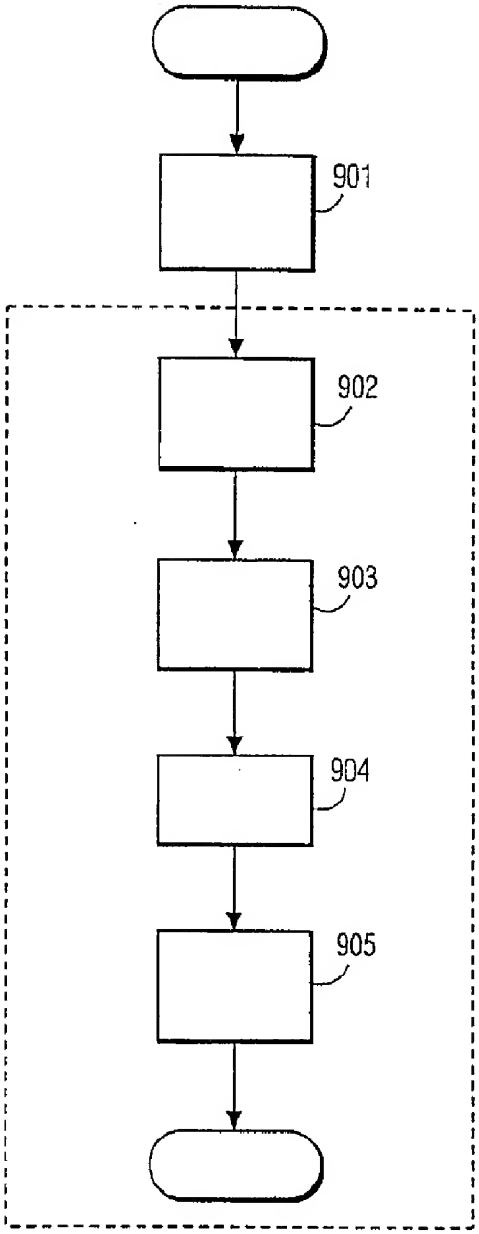
第７図

【図８】



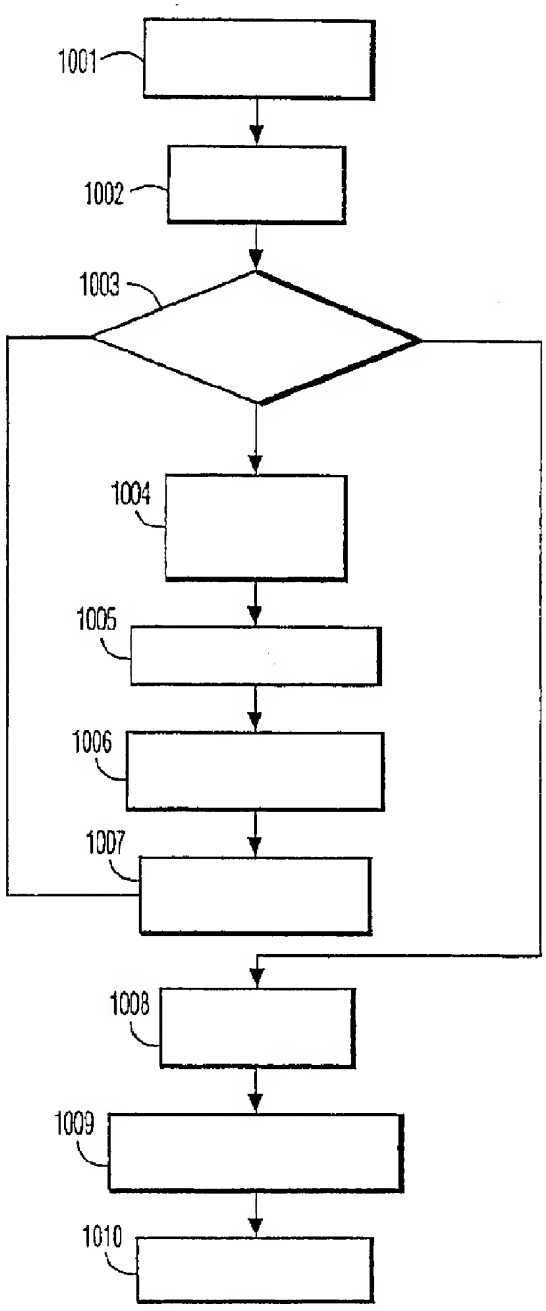
第８図

【図９】



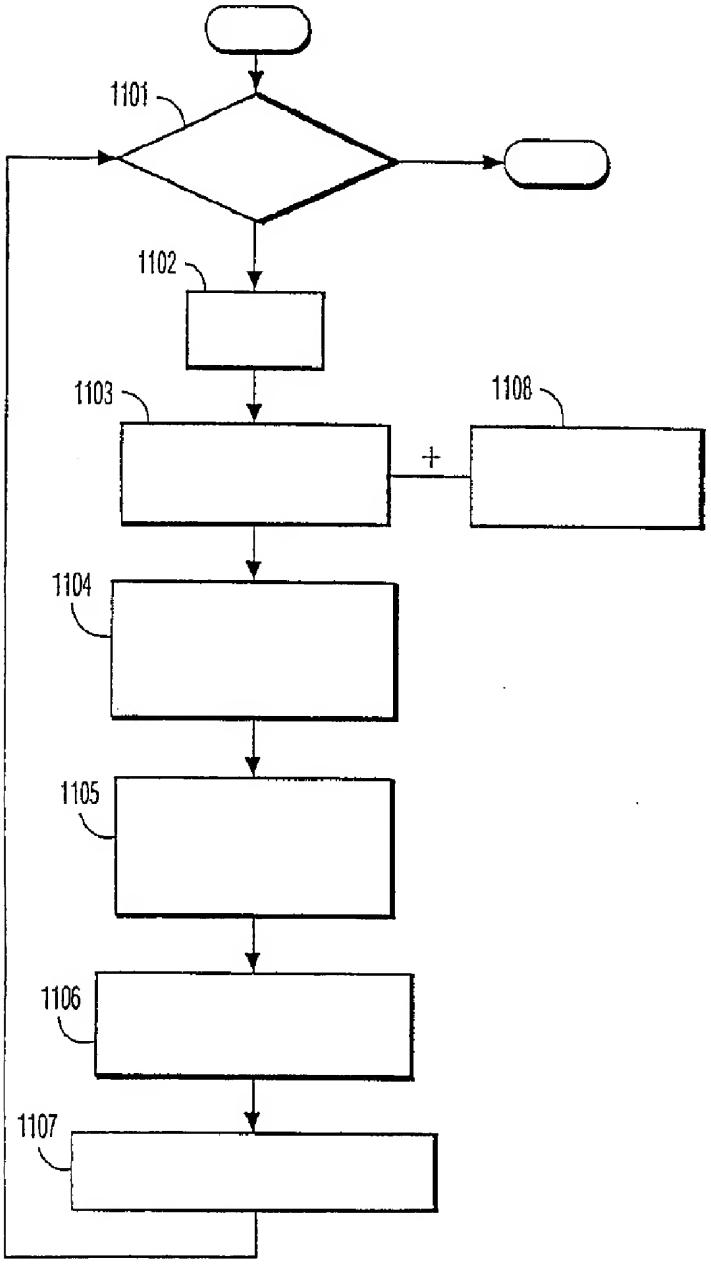
第９図

【図１０】



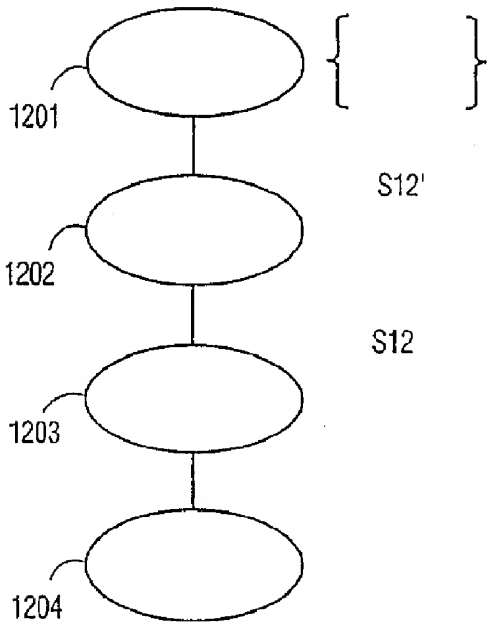
第10図

【図１１】



第11図

【図１２】



第12図

【国際調査報告】

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB 97/00558

A. CLASSIFICATION OF SUBJECT MATTER

IPC6: G06F 9/28, G06F 9/38 // G06F 12/04
According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC6: G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

SE,DK,FI,NO classes as above

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

WPIL, INSPEC, TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	Philips hopes to displace DSPs with VLIW: TriMedia processors aimed at future multimedia embedded apps. Brian Case, Microprocessor Report vol. 8, nr 16, p12(4) Dec 5, 1994 see whole document	1
A		2-17

X	WO 9519006 A1 (THE DOW CHEMICAL COMPANY), 13 July 1995 (13.07.95), page 17, line 6 - line 15, claims 1,13, abstract	1
Y		2,7
A		3-6,8-17
	--	

[X] Further documents are listed in the continuation of Box C. [X] See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" erlier document but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance: the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance: the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
27 November 1997	02.12.97
Name and mailing address of the ISA/ Swedish Patent Office Box 5055, S-102 42 STOCKHOLM Facsimile No. +46 8 666 02 86	Authorized officer Jan Silfverling Telephone No. +46 8 782 25 00

Form PCT/ISA/210 (second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

International application No.
PCT/IB 97/00558

C (Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 5179680 A (ROBERT P. COLWELL ET AL), 12 January 1993 (12.01.93), column 2, line 9 - line 39; column 6, line 17 - line 36; column 7, line 25 - line 44, claim 1, abstract, column 15, line 5 - line 15, column 15, line 55 - line 65	1
Y		2,7
A		3-6,8-17

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.
PCT/IB 97/00558

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 9519006 A1	13/07/95	CA 2180855 A	13/07/95
		EP 0739517 A	30/10/96
		US 5655133 A	05/08/97
US 5179680 A	12/01/93	US 5057837 A	15/10/91

Form PCT/ISA/210 (patent family annex) (July 1992)

フロントページの続き

(31)優先権主張番号　０８／６４９，７３３
(32)優先日　　1996年５月15日
(33)優先権主張国　　米国（ＵＳ）
(81)指定国　　ＥＰ(ＡＴ，ＢＥ，ＣＨ，ＤＥ，
ＤＫ，ＥＳ，ＦＩ，ＦＲ，ＧＢ，ＧＲ，ＩＥ，ＩＴ，Ｌ
Ｕ，ＭＣ，ＮＬ，ＰＴ，ＳＥ)，ＪＰ
(72)発明者　ハムパプラム　ハリ
オランダ国　5656　アーアー　アインドー
フェン　プロフ　ホルストラーン　６
(72)発明者　リー　エン　シー
オランダ国　5656　アーアー　アインドー
フェン　プロフ　ホルストラーン　６